Universidade de Aveiro
2016
Departamento de Economia, Gestão, Engenharia Industrial e Turismo

Nuno Augusto Simões Ferreira

Metodologia DevOps e suas mais valias para as Organizações

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia e Gestão Industrial, realizada sob a orientação científica da Doutora Ana Maria Pinto de Moura, Professora Auxiliar do Departamento de Economia, Gestão, Engenharia Industrial e Turismo da Universidade de Aveiro

O júri

Presidente

Prof. Doutora Maria João Machado Pires da Rosa
Professora Auxiliar, Universidade de Aveiro

Prof. Doutora Ana Maria Pinto de Moura
Professora Auxiliar da Universidade de Aveiro

Prof. Doutor Nuno Alexandre Pinto da Silva
Professor Coordenador do Instituto Superior de Engenharia do Porto

Agradecimentos

O principal agradecimento vai para os meus pais. Mãe e Pai, os meus heróis que me trouxeram até aqui, que me apoiaram e guiaram ao longo de todo o meu percurso, tendo eles contribuído imenso para hoje aqui chegar. É à vossa luta diária, ao vosso esforço, à vossa dedicação sem nunca pestanejar. A vocês vos devo tudo aquilo que sou. Sem vocês não estaria a entregar este documento e certamente não estaria a referir muitas das pessoas que à frente irei agradecer. O vosso filho será sempre vosso menino.

À minha Namorada, que me acompanhou, motivou, acreditou, aguentou e não me deixou nunca desanimar. Por ter lutado, acompanhado, alertado, preocupado, estrebuchado mas essencialmente por ter sido quem ela é. Por acreditares em mim.

À minha Madrinha, que muitas boleias me deu, muito aturou, pela disponibilidade total e por toda a atenção e dedicação. A ela lhe devo também um agradecimento com muito carinho.

A toda a minha família. Família essa de onde vim e onde cresci. Quem muito me ensinou e apoiou. Ao meu Padrinho, Irmãos, Tios, Tias, Primos e Avós. Todos sem exceção.

À Professora Ana Moura, minha orientadora de tese, que teve sempre a maior das atenções, disponibilidade e disposição, tendo em conta todas as dificuldades e limitações que tive para elaborar o presente documento.

À Deloitte Consultores, S.A. e suas pessoas, por todo o apoio e disponibilidade prestada. Em especial à Vânia Caetano, minha orientadora da empresa e que tanta paciência e dedicação teve para me ajudar a ultrapassar esta fase.

A todos os meus amigos que de uma forma ou de outra estiveram comigo ao longo destes anos. Ao André Couto, ao Roldão, ao Diogo e ao Miguel por estarem sempre quando foi preciso e que foram e serão sempre parte significativa daquilo que levo comigo daqui. Ao Tiago Alves e ao Pedro Lages por terem acreditado e puxado por mim, por terem assumido um papel muito importante para o meu desenvolvimento enquanto pessoa e cidadão, mas essencialmente pela amizade que tanta satisfação me deu e assim espero continuar a dar. A vocês os 6, obrigado por serem vocês mesmos.

Ao Paulo Alves e ao Daniel Oliveira, por também eles terem acreditado em mim e incentivado a iniciar a minha participação associativa estudantil, que tantas alegrias me trouxe.

A todas as pessoas que fizeram parte do meu percurso associativo na Associação Académica da Universidade de Aveiro, período esse que me irá marcar para o resto da vida. À Ana Duarte pela dedicação e entrega. Ao Gustavo Teixeira pela amizade e pelo compromisso que ainda hoje guardo por ele.

Palavras-chave DevOps, Software, Sistemas de Informação, Lean, Agile, Deployment.

Resumo

Num mundo em constante mudança, onde a tecnologia assume cada vez mais um papel central nas nossas vidas, as organizações são constantemente desafiadas a adaptarem-se de forma cada vez mais rápida e precisa às alterações de mercado.

No presente contexto a metodologia DevOps surge, assumindo-se como "upgrade" à metodologia Agile já existente e colocando o foco na criação mais rápida e eficiente de valor acrescentado para o negócio das organizações, ao longo de todo o ciclo de desenvolvimento.

Assim, o presente estudo procura elaborar uma revisão bibliográfica do seu conceito, apresentar as suas linhas de orientação, bem como confrontar as mais-valias para as organizações citadas com as encontradas em 18 casos de estudo da aplicação de soluções no âmbito da metodologia.

O objetivo final prende-se com a compreensão teórica da metodologia e do modo como esta pode ajudar as organizações a competir neste mundo cada vez mais rápido e complexo.

keywords

DevOps, Software, Sistemas de Informação, Lean, Agile, Deployment.

abstract

In a constant change world, where technology assumes a greater role on our life, the organizations are been always challenged to a faster and precise adaptation.

On this context the DevOps methodology also assumes itself as an Agile upgrade, looking to create value faster and efficient to the business through the development life cycle.

The scope of this study is to present a bibliographic revision of its concept, its guidelines and try to understand if the benefits of its implementation reported in 18 case studies are coincident with those identified on the bibliographic revision.

The final goal is to theoretically understand its concept and its real value for the organizations to compete in this faster and complex world.

Índice

Índice de Tabelas

Índice de Figuras

Capítulo 1 - Introdução

1.1. Introdução

O presente documento foi elaborado no âmbito da conclusão do Mestrado em Engenharia e Gestão Industrial. O seu foco centrou-se no estudo do conceito e impactos resultantes da metodologia DevOps, orientada para o desenvolvimento de *software*.

Hoje, todos temos consciência que grande parte daquilo que sustenta a nossa sociedade, quer seja nas empresas, no estado ou até no nosso dia-a-dia, são sistemas de informação. Quer seja na indústria automóvel, financeira, de comunicações, de transportes ou outras, cada vez mais são utilizados sistemas de informação que permitem às empresas gerir a sua atividade a todos os níveis: na interação com o cliente, na gestão de recursos, atividades e até mesmo incorporados nos seus próprios serviços ou produtos. A importância destes sistemas é portanto inegável, e face à sua importância o seu desenvolvimento é cada vez mais alvo de grande atenção, tendo em conta os custos e recursos que hoje esta atividade absorve dentro das organizações.

Todos estes pontos comprovam a importância que as sucessivas metodologias de desenvolvimento de *software* têm para as organizações. Estas procuram elencar conjuntos de boas práticas com vista à diminuição de erros de desenvolvimento, à maior rapidez e consequente diminuição custo, procurando garantir o foco no propósito de desenvolvimento e respetiva qualidade.

Estes erros, falhas de entrega, ou inadequação dos sistemas face ao pretendido, podem representar custos colossais para as organizações que neles investem. Basta pensar que em média são gastos, num projeto de desenvolvimento de *software* nos Estados Unidos, $2.322.000 por empresas de grande dimensão e $434.000 por empresas de pequena dimensão, sendo que no seu conjunto a economia americana gasta, por ano, mais de $250 mil milhões de dólares, e percebemos o impacto que falhas ou ineficiências no processo podem representar numa organização[1].

Posto isto, após várias metodologias de desenvolvimento de *software*, surge em 2008 o DevOps como metodologia que vem complementar a ainda predominante metodologia Agile,

nomeadamente cobrindo áreas ignoradas por esta última, procurando aumentar ainda mais a rapidez e eficiência das organizações nos seus projetos ou atividades de desenvolvimento. O presente estudo procura investigar as suas origens, princípios, entender o que vem acrescentar de novo às metodologias existentes e perceber as suas mais-valias, recorrendo simultaneamente a um conjunto de casos de estudo e à revisão bibliográfica do tema.

1.2. Questões da Investigação

O foco do presente estudo centra-se na metodologia DevOps, sendo que a principal questão que se pretende ver respondida é a seguinte:

- Os benefícios aos quais a metodologia DevOps se propõe, são realmente verificados após a sua adoção?

No entanto, para entender como estes benefícios são concretizados e o que podem significar para as organizações, torna-se também necessário responder às seguintes questões:

- Quais são, efetivamente, os benefícios a que o DevOps se propõe?
- Qual o seu conceito e linhas de orientação?
- De que forma pode contribuir para o sucesso das organizações?

1.3. Objetivos

Respondendo às questões atrás colocadas, é possível compreender:

- A metodologia DevOps, os seus componentes, ferramentas e raio de ação;
- A forma como esta vem complementar metodologias anteriores e como se diferencia destas;
- As mais-valias de negócio que a sua adoção poderá representar para as organizações;

- E se os resultados obtidos, após a sua adoção, correspondem às expectativas que lhe estão associadas;

1.4. Estrutura

O presente estudo está organizado em 5 capítulos, procurando criar uma sequência lógica que permita uma melhor compreensão dos conceitos e ideias apresentadas.

No **Capítulo I – Introdução**, está presente o âmbito do trabalho, bem como as questões e objetivos que lhe foram traçados, e a apresentação da estrutura do documento.

Após este, o **Capítulo II – Revisão Bibliográfica** procura fazer uma abordagem às principais metodologias de desenvolvimento que antecedem e servem de base à metodologia DevOps: as metodologias *Lean Software Development* e *Agile Software Development*.

Finda a revisão bibliográfica, surge o **Capítulo III – Conceito e Metodologia DevOps** cujo principal propósito é o de fazer uma introdução à metodologia, compreender as práticas que lhe estão associadas, explicar como esta vem complementar as metodologias que tem por base, e identificar os resultados a que se propõe atingir, bem como os seus maiores desafios.

No **Capítulo IV – Casos de Estudo da sua Aplicação** é caracterizada a amostra de casos de estudo recolhidos, explicada a metodologia de recolha de dados e consequente análise, demonstrados os resultados e, por fim, elaborada uma análise destes face aos conceitos introduzidos no capítulo 3.

Por fim, o **Capítulo V – Conclusões Finais** pretende descrever as principais conclusões do estudo face às questões e objetivos definidos, com base nos resultados e elações retiradas do capítulo anterior, sendo também apresentadas as limitações do mesmo e consequentes perspetivas de investigação futura.

Capítulo 2 - Revisão Bibliográfica

O *DevOps* enquanto metodologia de desenvolvimento e produção de *software* não surge de forma isolada. Para entender o seu conceito torna-se imperativo elaborar uma introspetiva às metodologias que lhe antecederam e que em certa medida lhe servem de base.

Os princípios e práticas da metodologia DevOps têm, por base as metodologias *Lean Software Developement* (ora em diante designado por Lean) e *Agile Software Development* (ora em diante designado por Agile)[2], que desde a década de noventa assumiram um papel cada vez mais preponderante no desenvolvimento de *software*.

No presente capítulo, pretende-se dar uma introdução a estas duas metodologias, procurando perceber como se relacionam e servindo assim de base para a abordagem da metodologia *DevOps*, permitindo entender melhor todas as suas vertentes.

2.1. Lean Software Development

O *Lean* tem como principal foco a adoção de um conjunto princípios e práticas de gestão de recursos, quer sejam eles materiais ou humanos, para assim alcançar os objetivos em baixo citados:

"Lean producers, on the other hand, set their sights explicitly on perfection: continually declining costs, zero defects, zero inventories, and endless product variety."

[3]

Num artigo intitulado de "*Principles of Lean Thinking*"[4], Poppendireck procura fazer uma comparação entre a metodologia *Lean* aplicada à manufatura e a aplicada ao desenvolvimento de *software*, fazendo um paralelismo entre os sete tipos de desperdício, identificados por Taiichi Ohno, com desperdícios correspondentes num processo de desenvolvimento de *software*.

Desperdícios na Indústria Tradicional	Desperdícios no Desenvolvimento de Software
Sobreprodução	Funcionalidades extra.
Inventário	Requisitos
Passos de processamento adicionais	Passos adicionais
Movimento	Procurar informação.
Defeitos	Defeitos não identificados pelos testes
Espera	Esperar, incluindo clientes.
Transporte	Handoffs

Tabela 1 - Paralelismo entre os desperdícios industriais e os de desenvolvimento de software[4].

De acordo com David J. Anderson[5], o conceito de *Lean* foi lançado pela primeira vez numa conferência no ano de 1992, em Estugarda, Alemanha, tendo sido sugerida a criação do conceito pouco tempo depois por Robert "Bob" Charlet em 1993.

No livro "*Lean Software Development – an Agile ToolKit*"[6] fazem menção ao conceito da metodologia *Lean* aplicada ao desenvolvimento de *software,* assumindo esta como sendo uma "ferramenta" a incluir no seio da metodologia Agile, abordada no subcapítulo seguinte. No livro podemos encontrar referências aos princípios base pelos quais a metodologia Lean se baseia. São eles[6]:

1. **A eliminação do desperdício**: no fundo a eliminação do desperdício aqui refere-se à eliminação de atividades que representem pouco ou nenhum valor acrescentado.

2. **A difusão do conhecimento adquirido**: permitir que eventuais erros, alterações ou pedaços de código pré construídos, sejam constantemente partilhados para que possam servir de ferramentas para futuros ou atuais desenvolvimentos.

3. **Tomar decisões tão tarde quanto possível**: Ao atrasar decisões estamos a assegurar que estas sejam tomadas com base na maior quantidade de informação possível, reduzindo deste modo a vertente especulativa (ainda que esta esteja sempre presente). Ao diminuir a especulação na tomada de decisão está-se a diminuir os riscos de futuras alterações de requisitos que representem impactos negativos para todo o desenvolvimento.

4. **Entregar o mais rápido possível**: Entregar peças de *software* rapidamente ao cliente, resulta também num feedback mais rápido e atualizado das suas necessidades, permitindo uma maior aproximação e uma adaptação constante mais eficiente.

"Without speed, you cannot delay decisions. Without speed, you do not have reliable feedback. (…) Speed assumes that customers get what they need now, not what they needed yesterday"

[6]

5. **Conceder poder de decisão à equipa**: As decisões quanto ao trabalho que tem de ser feito, como e quando, devem ser concedidas à equipa que está no "terreno" diariamente [6]. Muitas vezes o trabalho é organizado por uma equipa de gestão que não estando no terreno não tem o contacto necessário para a tomada de decisões acerca do trabalho a realizar.

6. **Incutir integridade naquilo que é desenvolvido:** Ao manter uma elevada integridade nos desenvolvimentos efetuados, está-se a garantir que estes comportam altos níveis de usabilidade, adaptados ao seu propósito, facilmente mantidos, adaptáveis e flexíveis [6].

7. **Atenção ao todo:** Sistemas de informação pressupõem um conhecimento profundo em muitas áreas. Um dos maiores problemas com o desenvolvimento é que os especialistas em cada área têm tendência a maximizar a performance da sua área em específico, em vez de se focarem na performance global do sistema.

2.2. Agile Software Development

Em 2001, dezassete "*developers*" reuniram-se para a criação da metodologia Agile, focada no desenvolvimento de *software*. Esta nova metodologia teve por base um conjunto de práticas tais como o "*Extreme Programing*", o "*Adaptive Software Development*", entre outras técnicas frequentemente apelidadas de "*Ligthweigth*" dada a existência de poucas regras, contribuindo deste modo para uma maior agilidade face a um mundo em constante mudança[7].

Após esta primeira reunião, e no período que se seguiu, foram posteriormente definidos quatro principais valores sobre os quais todos concordaram que o Agile deveria assentar. São eles[8]:

- Indivíduos e interações mais do que processos e ferramentas;
- *Software* funcional mais do que documentação abrangente;
- Colaboração com o cliente mais do que negociação contratual;
- Responder à mudança mais do que seguir um plano.

Após a definição destes quatro valores basilares é então, com base nestes, definido o "Manifesto Agile" onde estão patentes os doze princípios pelos quais a metodologia dever-se-á reger. São eles[8]:

1. A maior prioridade é, desde as primeiras etapas do projeto, satisfazer o cliente através da entrega rápida e contínua de *software* com valor.
2. Aceitar alterações de requisitos, mesmo numa fase tardia do ciclo de desenvolvimento. Os processos ágeis potenciam a mudança em benefício da vantagem competitiva do cliente.
3. Fornecer frequentemente *software* funcional. Os períodos de entrega devem ser de poucas semanas a poucos meses, dando preferência a períodos mais curtos.
4. O cliente e a equipa de desenvolvimento devem trabalhar juntos, diariamente, durante o decorrer do projeto.
5. Desenvolver projetos com base em indivíduos motivados, dando-lhes o ambiente e o apoio de que necessitam, confiando que irão cumprir os objetivos.

6. O método mais eficiente e eficaz de passar informação para e dentro de uma equipa de desenvolvimento é através de conversa pessoal e direta.
7. A principal medida de progresso é a entrega de software funcional.
8. Os processos ágeis promovem o desenvolvimento sustentável. Os promotores, a equipa e os utilizadores deverão ser capazes de manter, indefinidamente, um ritmo constante.
9. A atenção permanente à excelência técnica e um bom desenho da solução aumentam a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho que não é feito – é essencial.
11. As melhores arquiteturas, requisitos e desenhos surgem de equipas auto-organizadas.
12. A equipa reflete regularmente sobre o modo de se tornar mais eficaz, fazendo os ajustes e adaptações necessárias.

Como facilmente se constata, existem semelhanças entre estes doze princípios e os sete enunciados por Poppendieck & Poppendieck para o Lean. Peguemos por exemplo no primeiro princípio do Lean e facilmente constatamos que este é a concretização do décimo princípio do manifesto Agile. A simplicidade, ou como é referido a capacidade de diminuir a quantidade de trabalho realizado, é em si mesma a realização do princípio da eliminação do desperdício, reduzindo a cadeia de atividades a realizar ao restritamente necessário para o atingir dos objetivos, eliminando ou reduzindo deste modo atividades redundantes ou desnecessárias, ou seja, que não acrescentem valor[9].

A difusão do conhecimento adquirido, segundo princípio do Lean, está também patente no segundo, quarto e sexto princípio do manifesto Agile. Ao aceitar alterações aos requisitos em qualquer fase do desenvolvimento, como meio de adaptação constante às mudanças do ambiente de negócio para o qual se destina a solução, torna-se necessária uma partilha de feedback e conhecimento constante entre as diferentes partes envolvidas no processo de desenvolvimento[9].

Já o trabalho e contacto constante entre a equipa de desenvolvimento e o cliente, patente no quarto princípio do manifesto Agile, é por si só um princípio que procura aumentar a partilha de conhecimento e proporcionar as ocasiões necessárias para tal. O sexto princípio do manifesto Agile dá a indicação de como a comunicação e partilha de conhecimento deve ser feita, assumindo o contacto direto e pessoal como um canal preferencial[9].

Olhando para o terceiro princípio do Lean citado atrás, não encontramos uma tradução direta nos princípios explanados pelo manifesto Agile. No entanto este funciona como pré-requisito para o segundo princípio do manifesto. A prática de tomar decisões tão tarde quanto possível permite que eventuais alterações aos requisitos sejam mais facilmente incorporadas. Com o adiar das decisões até ao último momento é possível diminuir o risco da tomada de decisões que dificultem ou impossibilitem a adoção e incorporação futura de eventuais alterações de requisitos ao longo do processo[9].

No quarto princípio do Lean é possível encontrar uma correspondência quase direta ao primeiro do manifesto Agile. Aqui ambos encaram a rapidez de desenvolvimento e entrega como um ponto fulcral[9].

O quinto princípio do Lean corresponde quase diretamente ao quinto do Agile, já que ambos encaram a atribuição da responsabilidade de gestão e de decisão à equipa. Isto como forma de motivar e deixar as decisões a quem está diariamente no terreno e que por isso tem, em princípio, um conhecimento mais aprofundado dos temas[9].

Posto isto, é possível concluir que a metodologia *Agile* utiliza ou pode talvez mesmo pressupor a utilização dos princípios e valores da metodologia Lean. Esta analogia é genericamente aceite, sendo que Perterson[9] elaborou uma análise comparativa destas duas metodologias, tendo chegado às seguintes conclusões:

- Ambas apontam o seu foco no cliente;
- Os princípios de ambas são similares e partem de bases semelhantes;
- Ambas mantêm ainda assim princípios próprios únicos (exemplo do sétimo principio do Lean);
- A metodologia Agile, ao contrário da metodologia Lean, aponta para diferentes abordagens ao processo de desenvolvimento;

Olhando para estes quatro pontos, podemos afirmar que os primeiros três foram já comprovados na presente análise, ao comparar os princípios associados às duas metodologias. O quarto ponto será alvo de análise na secção seguinte, 2.2.1.

2.2.1. Abordagem Processual da Metodologia Agile

Se atendermos aos princípios da metodologia Agile, transcritos do manifesto, podemos constatar que estes implicam não só uma alteração à abordagem como se gere o processo de desenvolvimento, como também uma alteração ao modo como este processo deve decorrer.
As conclusões de Peterson (2010) apontam também elas no mesmo sentido, pelo que no presente capítulo pretende-se elencar a abordagem da metodologia Agile ao processo de desenvolvimento, utilizando o modelo *Waterfall*, anterior em termos temporais, como termo de comparação.

No modelo *Waterfall* cada fase do processo (levantamento de requisitos, elaboração da arquitetura da solução, desenvolvimento, teste e entrega) apenas é iniciada quando a anterior estiver totalmente finalizada[10].

É exatamente neste ponto em que a metodologia Agile diverge ao "ser iterativa (realizando inúmeros ciclos até estar completo o desenvolvimento), incremental (não entrega uma solução inteira de uma só vez), e emergente (os processos, princípios e estruturas de trabalho são reconhecidas durante o projeto em vez de serem pré-determinadas) "[11]. Para além disto tudo, as diferentes fases (Definição de requisitos, desenvolvimento, testes, etc…) são executadas paralelamente. Existe portanto uma mudança não só de abordagem metódica de trabalho como também uma alteração à forma como o processo de desenvolvimento deverá ocorrer.

Peterson[9] afirma que a metodologia Agile, em comparação com abordagens tradicionais, tais como o desenvolvimento orientado ao planeamento (plan-driven development), tem como objetivo entregar continuamente software funcional que possa ser demonstrado ao cliente, ilustrando o último estado e averiguando se a solução entregue preenche as suas necessidades. Assim, os clientes podem dar feedback mais cedo e garantir que o que está a ser desenvolvido está em linha com as suas espectativas."

É através deste feedback contínuo, por parte do cliente, que é possível garantir um maior alinhamento do desenvolvimento em relação às necessidades do cliente. Este alinhamento resulta num menor risco de no momento da entrega final o sistema desenvolvido não esteja conforme o pretendido pelo cliente ou utilizador final, já que este foi sendo sucessivamente aprovado ao longo das diferentes iterações.

2.2.2. Como vem o DevOps colmatar o Agile

Tal como já tivemos oportunidade de constatar, a metodologia Agile foca-se na fase de desenvolvimento de software. Isto significa que deixa de lado a entrada em produção, tipicamente deixada à responsabilidade de uma equipa específica: as Operações.

É neste ponto que a metodologia *DevOps* surge, assumindo-se como um "*upgrade*" à metodologia Agile. Segundo um artigo da Forrester[12], o "DevOps aparece como sendo uma melhoria ao Agile, adicionando práticas que permitem processos altamente automatizados de entrega de software, ao mesmo tempo que providencia maior visibilidade e mais pontos de controlo".

Manish Virmani[13], Gestor de Desenvolvimento da IBM Rational, refere que "com o passar dos anos as organizações tem adotado muitas otimizações nas suas práticas de desenvolvimento de *software* (transformação agile). No entanto, ao longo de toda esta evolução o foco tem sido maioritariamente no desenvolvimento, deixando de lado a vertente de operações...".

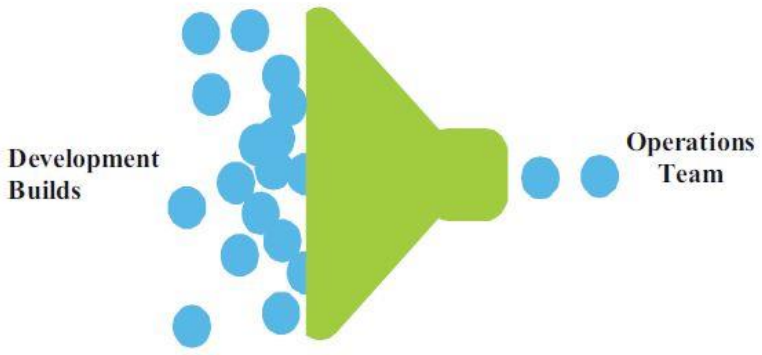


Figura 1 - Gargalo entre as equipas de desenvolvimento e operações[13].

Segundo um artigo na Scrum Aliance, a metodologia Agile ao focar-se apenas no aumento da rapidez com que o desenvolvimento é efetuado, deixando de fora as equipas de operações, cria um gargalo no processo. Ou seja o processo de desenvolvimento é de facto acelerado, no entanto as equipas de operações não conseguem acompanhar esse aumento de velocidade, criando um gargalo onde os componentes desenvolvidos acabam por esperar para entrar em produção[14].

Como facilmente percebemos, a criação deste gargalo significa que apesar da maior velocidade com que é criado valor, este não tem impacto no negócio após a sua criação, acabando por ter de aguardar pela sua vez de entrada em produção, representando assim um claro desperdício.

Capítulo 3 - Conceito e Metodologia DevOps

Como já vimos, o DevOps vem complementar o Agile, ao focar-se num problema deixado por este na última fase de desenvolvimento de software.

De forma a resolver o gargalo abordado no capítulo anterior, o principal foco de ação do DevOps centra-se numa maior sincronização e comunicação entre as equipas tipicamente envolvidas nesta fase do desenvolvimento - as equipas de desenvolvimento e operações - de modo a garantir que o momento de passagem para produção seja feito de forma mais simples e rápida. O próprio nome "*DevOps*", inicialmente lançado por Patrick Debois e Andrew "Clay" Shafer em 2009, é o resultado da junção de Dev – Desenvolvimento – e Op – Operações[15].

Assim, o presente capítulo pretende dar uma visão global da metodologia, abordando a relação entre estas equipas – secção 3.1. -, uma framework conceptual da mesma – secção 3.2. -, a forma como encara o ciclo de desenvolvimento e respetivas práticas tipicamente associadas – secção 3.3. -, a tecnologia que lhe está tipicamente associada – secção 3.4 - e por fim os desafios que se lhe impõem, bem como as respetivas mais-valias – secções 3.5. e 3.6., respetivamente.

3.1. Equipas de Desenvolvimento e Operações

A equipa de desenvolvimento (Dev), como o próprio nome indica, é a equipa responsável pelo desenvolvimento de novas peças de *software*. Durante este período, a solução é levantada, desenhada, concebida e testada (quer sejam testes unitários, funcionais, de performance, entre outros) em ambientes de desenvolvimento, teste, pré-produção, e outros que possam existir conforme necessidade.

Após a sua conceção e respetivos testes, a solução é colocada em produção, ou seja em ambiente de negócio real, onde interage diretamente com o negócio, sendo aí que desempenha o seu verdadeiro propósito. A entrada em produção, marca o momento de interação entre a equipa de desenvolvimento e de operações, responsável pela entrada em produção do código/solução desenvolvida e respetiva manutenção.

Tendo em conta a natureza e tipologia de funções de cada uma destas equipas, é fácil de perceber o porquê da existência de alguns "conflitos" quando estas interagem. Tais conflitos estão essencialmente relacionados com os diferentes focos de ambas as equipas.

A equipa de desenvolvimento, é a equipa cujo foco está em garantir a criação de novas soluções, para que os sistemas se adaptem às alterações das necessidades de negócio das organizações. Por outro lado, a equipa de operações encara estas novas soluções como o seu maior "inimigo", dado que representam a principal ameaça ao seu propósito: o de garantir a estabilidade e fiabilidade do sistema que se encontra em produção, e consequentemente do negócio da organização.

Michael Hiitterman, no livro "DevOps for Developers" da sua autoria, refere que a metodologia DevOps pressupõe uma abordagem de equipas transversais, trazendo o Agile também para as operações. O objetivo é juntar as duas equipas numa só, partilhando ferramentas e práticas de trabalho, e pressupondo um conjunto de valores interpessoais, que promovam o trabalho em equipa[16]:

- Respeito mútuo entre todos;
- Compromisso à volta dos objetivos partilhados;
- Autoria coletiva;
- Partilha de valores.

Para que tal seja possível, Michael Hiitterman refere também a importância na forma como são geridas estas novas equipas, tendo em conta os seus diferentes focos de ação. Um dos exemplos que dá é a questão dos incentivos. O facto de por um lado se incentivar pessoas de desenvolvimento para a criação de soluções inovadoras, e por outro lado incentivar pessoas de operações para garantir a estabilidade do sistema, vai inerentemente contribuir para o aumentar de conflitos entre elas. A sua sugestão passa pela alteração na forma como estes são concedidos. O objetivo é que seja transversal a todos os elementos, dando incentivos não consoante as inovações ou as garantias de estabilidade, mas antes pelo número de soluções que ao mesmo tempo sejam inovadoras e estáveis.

Michael Hiiterman refere ainda que o DevOps leva a equipas com um conjunto de especialistas que partilham as suas capacidades e experiências, contribuindo para que todos tenham, pelo menos, um entendimento básico dos problemas uns dos outros.

Em suma, a metodologia DevOps procura juntar estas duas equipas – Dev e Ops – de modo a que, partilhando objetivos comuns, passem a trabalhar de forma mais sincronizada, a partilhar métodos e práticas de trabalho, e a partilhar conhecimento continuamente, permitindo assim o planeamento e desenvolvimento de soluções que sejam ao mesmo tempo inovadores e estáveis, servindo o propósito de todos.

3.2. Framework CALMS

Até aqui sublinhámos mais de uma vez o enfoque que o DevOps dá à interação entre as equipas de desenvolvimento e operações, no entanto isto não é realizado de forma ah-hoc e sem nenhum método. O Devops assenta num conjunto mais abrangente de linhas orientadoras, que têm vindo a ser resumidas dentro de frameworks conceptuais. Estas têm evoluído ao longo do tempo e o seu principal objetivo é o de demonstrar essas linhas orientadoras de forma simples, intuitiva e resumida.

No ano de 2010, John Willis e Damon Edwards, conceberam a primeira *framework* para a metodologia *DevOps*, com o acrónimo de CAMS[17]. Mais tarde, Jez Humble adicionou-lhe um "L", de *Lean*, ficando assim o acrónimo a denominar-se de CALMS.

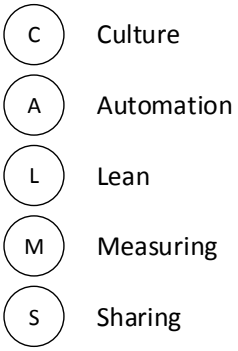


Figura 2 - Framework CALMS.

Nos próximos subcapítulos descrevem cada um destes pontos tendo por base uma publicação da Forrester sobre o tema[18].

3.2.1. Cultura

O principal desafio para as organizações face à cultura DevOps é o da passagem de uma organização estruturada em silos (departamentalizada) para uma organização transversal. Recordemo-nos que este foi o tema central referido no subcapítulo anterior para o melhoramento da relação entre Dev e Ops. No entanto, os seguintes princípios são apontados como sendo fundamentais para esta mudança cultural:

- A capacidade de falhar depressa sem ser desacreditado por isso, ou seja promover a deteção de falhas e erros o mais cedo possível, para assim combatê-los antecipadamente;

- A disponibilidade mental constante para experimentar novas metodologias, promovendo assim uma abertura a novas ideias que possam melhorar e tornar mais eficiente a forma como o trabalho é realizado.

- A abertura para sugestões de melhoramento por parte de qualquer um, em qualquer fase do processo de desenvolvimento;

- Espirito crítico e construtivo que questione constantemente o *status quo*;

- E uma atitude que promova a correção em vez da culpabilização.

3.2.2. Automação

Tipicamente, os processos de desenvolvimento de software são marcados por uma série de processos manuais, cujo resultado é o aumento do custo de desenvolvimento e entrega da solução, uma cadência reduzida do ciclo de vida das *releases* e uma qualidade inferior, face a erros humanos. O DevOps pressupõe a automatização desses processos manuais com vista ao aumento da velocidade, e respetiva qualidade, do que é desenvolvido ao longo de todas as fases do ciclo de desenvolvimento.

3.2.3. Lean

Tal como já aqui foi falado, o DevOps pretende continuar a usar o Agile no lado das Operações, de forma a obter *releases* mais pequenas e contínuas. De modo a manter este ritmo, todas as fases do ciclo de desenvolvimento terão de se tornar flexíveis e mais "leves" tanto quanto possível, nunca perdendo o foco na qualidade.

3.2.4. Medição e Controlo

Quando nos referimos à **medição e controlo** referimo-nos à importância de manter um acompanhamento próximo não só do estado atual como também de eventuais melhorias implementadas pela metodologia como, por exemplo, a automação. Sem a definição de métricas, como por exemplo o intervalo de tempo médio entre a identificação de um *bug* e a sua resolução em produção, ou entre a entrada de um novo requisito e a sua concretização, não podemos auferir acerca do real impacto das medidas tomadas. A identificação das métricas e consequentes medições são essenciais não só para entender o estado atual das coisas, como também para identificar focos de melhorias ou defeitos que sejam necessários manter ou corrigir no futuro.

3.2.5. Partilha

Por último, a **partilha** (*sharing*) assume aqui um papel preponderante. O conceito de partilha de informação surge em diversos âmbitos. Ao aumentar a velocidade e frequência com que as *releases* e *deployments* são feitos, o DevOps pretende que o utilizador final tome contacto o mais rapidamente possível com os desenvolvimentos efetuados, de forma a obter um *feedback* constante. Este *feedback* contante não é exclusivo apenas do utilizador final, mas destina-se também a todos os participantes nas diferentes fases do ciclo de desenvolvimento, sendo promovido através de uma maior comunicação e aproximação interna entre as diferentes pessoas, que é um dos focos principais da cultura DevOps.

3.3. Ciclo de Desenvolvimento de Software e Práticas Associadas

O aumento da rapidez de desenvolvimento e consonância com a manutenção da qualidade, que a metodologia DevOps procura atingir, é em grande medida alcançado por um conjunto de práticas de desenvolvimento que esta pressupõe que sejam aplicadas.

Para entender estas práticas, e em que fase têm lugar, torna-se necessário abordar primeiro o ciclo de desenvolvimento no seu todo. Este ciclo é composto por diversas fases que podem ser observadas na DevOps *delivery pipeline*, segundo Manish Virmani[13], na figura em baixo demonstrada:

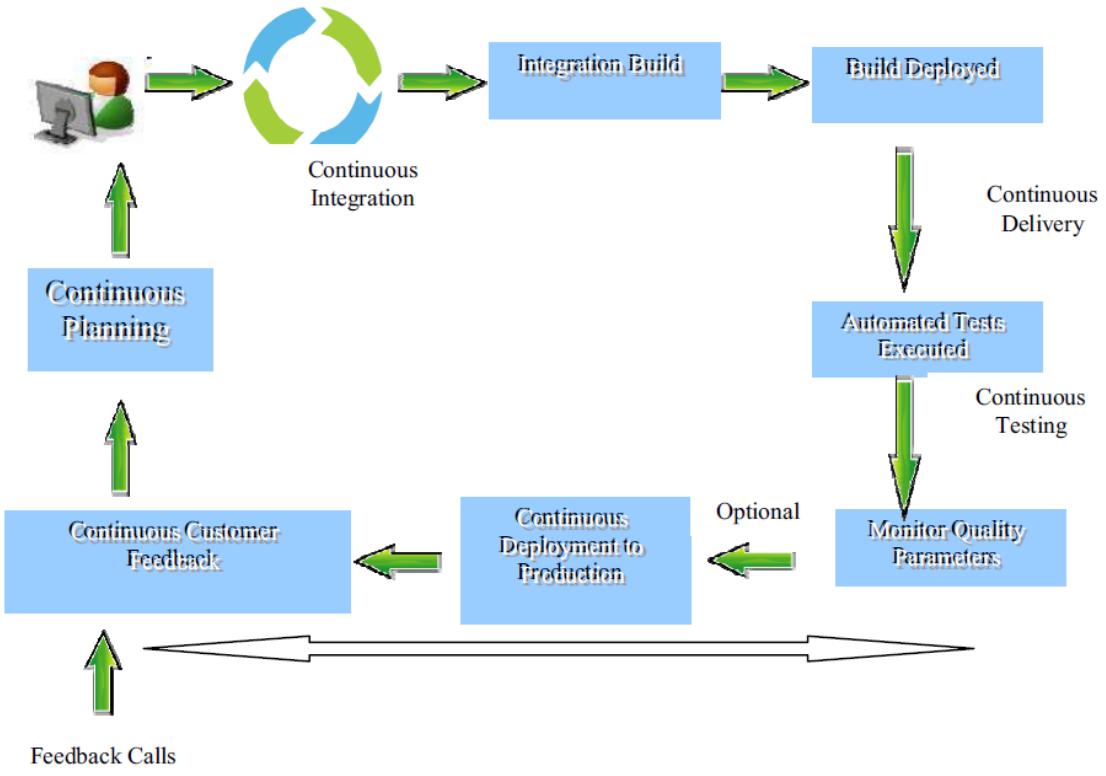


Figura 3 - Ciclo de desenvolvimento DevOps[13].

Este é o ciclo de desenvolvimento sobre a perspetiva DevOps e aqui é possível ter uma visão global do fluxo, observar todas as fases bem como a respetiva sequência. A metodologia pressupõe esta visão cíclica do processo de desenvolvimento, à qual lhe incorpora um conjunto de práticas, tal como é possível ver na imagem.

No total é possível identificar 7 práticas diferentes, no âmbito de desenvolvimento de software, sendo que cada uma destas desempenha um papel específico nas suas fases do ciclo correspondentes. São elas:

1. *Continuous Integration* (integração contínua de código);
2. *Continuous Testing* (testar continuamente);
3. *Continuous Delivery* (entrega contínua de software pronto a usar);
4. *Continuous Deployment* (entrada em produção contínua de código);
5. *Continuous Customer feedback* (feedback contínuo do cliente);
6. *Continuous Planning* (planeamento contínuo);
7. Continuous Monitoring (monitorização contínua).

Ao contrário do que se possa pensar, a maioria destas práticas de desenvolvimento não são exclusivas da metodologia DevOps, sendo que muitas antecedem a mesma e eram utilizadas no âmbito, por exemplo, da metodologia Agile.

Nos subcapítulos seguintes serão abordadas cada uma destas práticas de desenvolvimento. De sublinhar que a prática de *continuous testing* não é neste estudo abordada isoladamente, dado que, segundo a revisão bibliográfica efetuada, a prática de *continuous delivery* engloba/pressupõe esta mesma prática.

3.3.1. Continuous Integration

Segundo Sanjeev Sharma[2], a prática *continuous integration* consiste na integração contínua, ou frequente, do trabalho desenvolvido pelos diferentes *developers*.

Tradicionalmente, os *developers* desenvolvem o código, executam os testes unitários necessários e passam imediatamente para o próximo componente a desenvolver, sendo que os diferentes componentes desenvolvidos serão posteriormente integrados com todos os restantes[19].

Quando esta integração é feita em bloco, e não continuamente, existe um risco acrescido do surgimento de um maior número de erros e de estes impactarem todos os componentes já desenvolvidos, o que numa fase adiantada do desenvolvimento poderá representar atrasos significativos e eventuais custos adicionais[19].

A prática *continuous integration* pretende assim transformar a integração de código num processo contínuo, e não pontual. A Agile Alliance define os objetivos desta prática como sendo a minimização da duração e esforço necessários por cada integração a realizar, e a possibilidade de a qualquer momento apresentar uma versão preparada para entrar em produção (*release*)[20].

Assim, a prática de *continuous integration* promove a automação dos processos de "*build*", integração numa versão da solução e respetivos testes de aceitação, que procuram auferir a estabilidade e qualidade da solução após a integração da nova peça de *software*[20].

3.3.2. Continuous Delivery & Continuous Deployment

A abordagem conjunta destas duas práticas de desenvolvimento tem como objetivo fornecer uma distinção clara entre as duas, que frequentemente são referidas como sendo termos diferentes para a mesma prática[21].

Começando pela prática de *continuous delivery*, esta tem como principal objetivo a entrega contínua de valor ao cliente, sob a forma de versões prontas a entrar em produção a qualquer momento. No entanto, podemos facilmente questionar-nos acerca das razões para não colocar logo em produção toda e qualquer nova funcionalidade que vá sendo desenvolvida, assim que esta esteja integrada e testada[22].

Indústrias fortemente reguladas, como é o exemplo da indústria financeira, estão em constante mutação no que aos requisitos regulamentares diz respeito. Como é expectável, estas constantes alterações têm impactos nos sistemas que permitem hoje qualquer banco ou seguradora manter

a sua atividade. Neste tipo de indústrias a prática de *continuous delivery* adequa-se, dado que permite um estágio final da "release" antes da entrada em produção, permitindo assim a verificação por parte de auditores ou reguladores de mercado.

Esta é a principal diferença do *continuous delivery* face ao *continuous deployment*. A prática de *continuous deployment* pressupõe que sempre que um desenvolvimento é integrado e totalmente testado, este deve entrar imediatamente em produção[21]. Este processo de entrada em produção é, ao contrário do verificado na prática de *continuous delivery*, um processo automatizado, tal como é demonstrado na imagem abaixo.

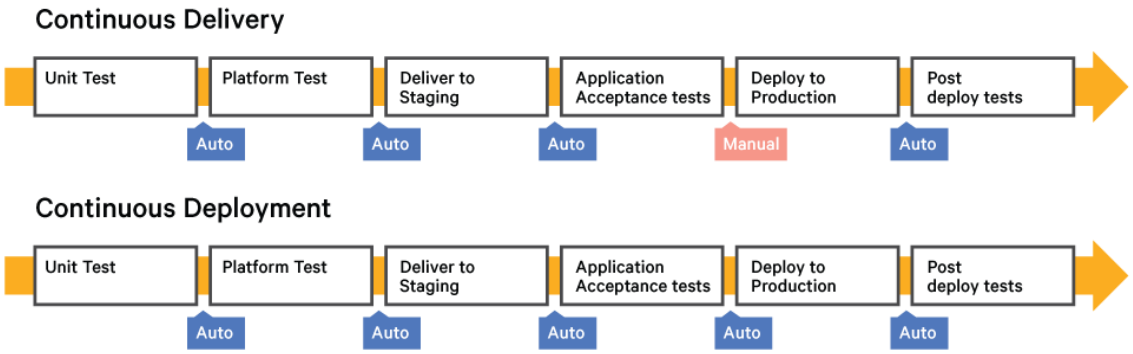


Figura 4 - Continuous Delivery vs Continuous Deployment[23].

3.3.3. Continuous Feedback

A prática de *continuous feedback* tem como conceito base a recolha contínua de feedback por parte do utilizador final, como forma de procurar garantir que as suas necessidades/aspirações foram ou venham a ser correspondidas da forma mais rápida e eficiente possível.

Para que se obtenha um feedback contínuo por parte do utilizador final, é necessário garantir a entrega contínua dos componentes, para que este possa tomar contacto e validar[24]. Esta validação pode ocorrer em dois momentos distintos: antes e após a entrada em produção, permitindo aumentar a confiança para a entrada em produção e a conformidade efetiva após a sua entrada[25].

Este feedback contínuo permite a qualquer negócio tornar-se mais ágil e pró-ativo na adequação às reais necessidades do utilizador, permitindo manter-se focado e a trabalhar nas questões que representam efetivamente interesse para o utilizador e que se materializem em valor acrescentado[2].

3.3.4. Continuous Monitoring

O objetivo da prática de *continuous monitoring* é, como o próprio nome indica, o de manter uma monitorização contínua sobre os diferentes processos ao longo do ciclo de desenvolvimento[2].

Com o conjunto de práticas já aqui abordadas, é possível recolher de modo também ele automático todo um conjunto de informação acerca dos diferentes processos, através da medição de um conjunto de métricas à partida definidas[13].

Mantendo uma visão em tempo real acerca do estado das diferentes fases, com a recolha automática de dados e métricas, é possível à gestão identificar problemas de modo mais rápido, preciso e simples, disponibilizando assim mais tempo para a definição e implementação de melhores soluções. Para além disso, esta visão permanente permite também definir e quantificar ricos com maior celeridade e precisão, possibilitando antecipar eventuais problemas.

3.3.5. Continuous Planning

Seja em desenvolvimentos de âmbito fechado ou contínuo, a prática de *continuous planning* é essencial para garantir uma constante e rápida adaptação às alterações das condições de mercado[13], ou ao feedback que vai sendo percecionado por parte do cliente[2] (tal como observado no subcapítulo anterior 3.3.3.).
No planeamento de tarefas e componentes a realizar, hoje em dia é comum aplicarem-se técnicas latentes do *lean thinking*, que pressupõem começar com um âmbito reduzido e ir indo acrescentado novas componentes a desenvolver, alargando o âmbito, à medida que vai sendo conhecido o feedback por parte do utilizador final daquilo que é desenvolvido[2]. Esta técnica vai de encontro ao terceiro princípio identificado no capítulo 2 para a metodologia Lean: tomar decisões tão tarde quanto possível.

Posto isto, o DevOps pressupõe e possibilita a execução da prática de *continuous planning* através da existência de um *product backlog,* priorizado conforme o feedback contínuo recolhido por parte dos utilizadores finais. Este feedback contínuo permite adaptar constantemente a priorização de atividades do *backlog*, garantindo assim que as prioridades de desenvolvimento estão em linha com as prioridades do negócio e que estas, por sua vez, estão também em linha com as necessidades e aspirações dos utilizadores finais[13].

3.4. Tecnologia na Metodologia DevOps

Tal como foi sucessivamente referido no subcapítulo anterior, as práticas de desenvolvimento inerentes à metodologia DevOps pressupõem a automação de processos. Para isso, é necessário que existam um conjunto de tecnologias que o permitam.

Existem hoje ferramentas para a gestão de *releases* ou para a automação dos *deployments*, que permitem o controlo e registo dos *deploys* feitos, em que pontos da infraestrutura e entre quaisquer fases do ciclo de desenvolvimento[2].

No entanto, para além de outras tecnologias existentes, a tecnologia *Infrastructure as a Code*, confere uma grande ajuda a ambas as equipas, Dev e Ops, no que diz respeito à gestão da infraestrutura que suporta todo o ciclo de desenvolvimento.

3.4.1. Infrastructure as a Code

De acordo com Sanjeev Sharma, tecnologia *Infrastructure as a Code* é considerada uma tecnologia essencial à metodologia DevOps, permitindo-lhe gerir a escala e velocidade com que os diferentes ambientes são provisionados e configurados, permitindo igualmente práticas como o *Continuous Delivery*[2].

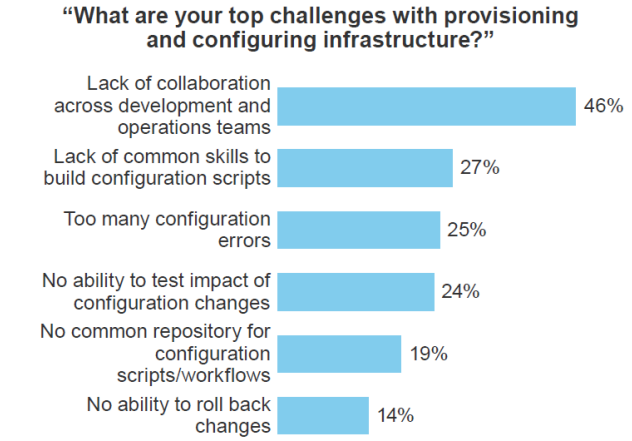
Ao longo do ciclo de desenvolvimento, abordado no subcapítulo 3.3, poderão existir vários ambientes, que mais não são do que servidores ou máquinas virtuais onde os desenvolvimentos são concebidos, testados ou executados. Quando, por exemplo, nos referimos à entrada em produção de uma *release*, ou de um pedaço de código desenvolvido, referimo-nos à passagem

desse desenvolvimento de um ambiente de teste, ou pré-produção, para um ambiente de produção.

Como também já tivemos oportunidade de ver através das práticas *continuous* abordadas no subcapítulo 3.3, um dos pressupostos metodologia DevOps é a automação da passagem de código entre as diferentes fases de desenvolvimento, integração, testes e finalmente entrega, podendo eventualmente entrar diretamente em produção. Posto isto, é essencial que todas as configurações (a nível de software, performance, network, entre outras) destes ambientes se encontrem o mais possível em perfeita sintonia com o ambiente de produção, que no fundo dita as condições finais já que é para ele que se destina todo o desenvolvimento. Ao garantir isto, estamos a minimizar o risco de eventuais erros relacionados com a eventual inaptidão do código desenvolvido para "correr" no ambiente ao qual se destina.

De acordo com um estudo elaborado pela Forrester acerca do tema[26], é referido que a tecnologia *Infrastructure as a Code* permite não só libertar a equipa de operações do provisionamento manual de ambientes, como também libertar as equipas de desenvolvimento e qualidade da espera para que estes estejam prontos, assegurando que estão sempre atualizados e "limpos".

Num outro estudo levado a cabo pela Forrester, com base num questionário a 300 profissionais de IT, na construção e *release* de *software*, foram encontrados os seguintes resultados para os principais desafios no provisionamento e configuração da infraestrutura[27]:



Base: 300 IT professionals involved with the build and release of software
Source: A commissioned study conducted by Forrester Consulting on behalf of Microsoft, February 2015

Figura 5 - Desafios no provisionamento e configuração de infraestrutura[27].

No mesmo estudo, foram também questionados outros 150 profissionais, que utilizam a tecnologia *Infrastructure as a Code*, acerca dos principais benefícios alcançados através da utilização da mesma[27]:

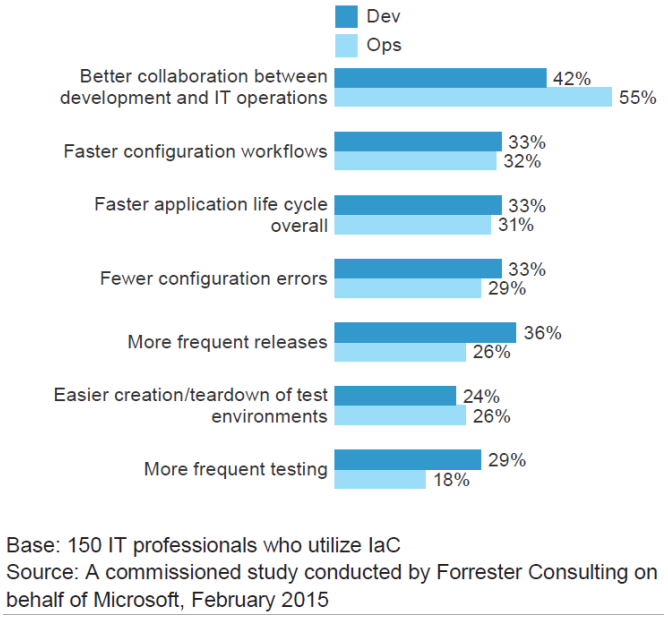


Figura 6 - Benefícios alcançados após a implementação da tecnologia Infrastructure as a Code[27].

3.5. Desafios à implementação do DevOps

Na implementação de uma qualquer metodologia, seja ela o Lean, Agile ou outra, existem sempre desafios. A metodologia DevOps não é exceção e tal como já foi possível observar nos subcapítulos anteriores existem mudanças de cariz tecnológico e cultural, que por vezes podem implicar alterações profundas na estrutura e processos de uma organização.

Jim Bird[28], em "DevOps for Finance", enunciou e descreveu os principais desafios à implementação do DevOps na indústria financeira. A escolha desta publicação, como principal referência, deve-se ao facto da mesma abordar os desafios de implementação numa indústria cujas principais referências são a sua elevada complexidade, dimensão, regulamentação, e elevada taxa de utilização de sistemas de informação para a gestão do negócio. São eles:

- Problemas organizacionais;
- O custo elevado em caso de falha;
- A complexidade e interdependência dos sistemas existentes;
- Os custos de "*compliance*";
- Ameaças de segurança.

Como principias **problemas organizacionais**, Jim Bird cita um artigo de Rachel Shannon-Solomon onde a grande dimensão e complexidade das empresas representam problemas, ao significar estruturas organizadas em silos (quando a metodologia DevOps pressupõe exatamente o oposto) e uma maior apetência para uma inércia organizacional na adoção de novas práticas.

Estes problemas são preponderantes tendo em conta que, tal como também citado, representam não só um desafio à dimensão do trabalho de implementação, como à dificuldade da mudança cultural que a metodologia DevOps pressupõe. No entanto, Jim Bird refere exemplos de sucesso em grandes organizações como a Google e a Amazon, apontando como fatores críticos de sucesso a vontade e visão da gestão e o talento dos recursos humanos em cenários de grande escala.

A questão do **custo elevado em caso de falha** é abordada por Jim Bird segundo duas perspetivas: a perda de receita e a eventual perda de confiança por parte do cliente. Ambas são apontadas como as principais consequências no caso de um erro que coloque em baixo os sistemas que gerem o negócio. Jim Bird dá como exemplo a falha dos sistemas da NASDAQ aquando da entrada em bolsa do Facebook em 2012, que representaram não só um elevado custo para a organização e investidores como também a sua descredibilização.

Outro exemplo dado, fora da indústria financeira, foi o da Amazon que quando teve os sistemas em baixo durante 30 minutos, perdeu cerca de $66,240 por minuto. No fundo este ponto mais não é do que um risco, sensível, que é visto por muitos como um desafio à implementação da metodologia DevOps, quando o seu mote é, segundo Jim Bird, "*failing fast and failing early, learning into failure, and celebrating failure*". Quando conciliamos este mote à dimensão e

sensibilidade do risco que acabámos de falar, principalmente numa indústria como a financeira, é fácil perceber a relutância presente em muitos líderes.

A complexidade e interdependência dos sistemas podem também representar um enorme desafio para a metodologia. Como é facilmente percetível, em sistemas complexos torna-se por vezes extremamente complicado rastrear os erros até à sua fonte e a própria antecipação dos mesmos torna-se muito mais complicada.

Outra questão identificada está relacionada com questões de **"*compliance*"**. Jim Bird refere as questões regulamentares como um desafio preponderante para uma metodologia cujo mote já aqui foi citado. Em indústrias fortemente reguladas, esta questão pode efetivamente representar um obstáculo para uma implementação DevOps, dada a obrigatoriedade da existência de extensivas auditorias ao que é colocado em produção, de evidências relativamente às responsabilidades e consciência face às decisões que são tomadas.

Um dos exemplos citados por Jim Bird está relacionado com a separação de papéis ao longo do ciclo de desenvolvimento. Os auditores e reguladores, por norma, exigem a separação clara de responsabilidades e papéis por cada um dos intervenientes no ciclo de desenvolvimento, como forma de garantir que ninguém possui o poder absoluto ao longo do ciclo e que questões como a integridade e confidencialidade de dados são respeitadas. Posto isto, Jim Bird refere que a metodologia DevOps, ao pressupor a destruição de silos organizacionais e criação de equipas transversais, com partilha de responsabilidades, poderá em certos casos chocar com esta questão.

Por fim, Jim Bird aponta a questão da **segurança** como outro dos desafios à implementação da metodologia. No entanto, como o próprio refere, o desafio não é se o DevOps não responde de modo eficaz ao combate e prevenção de ameaças de segurança, mas sim o preconceito que equipas responsáveis pelo *compliance* e segurança podem ter relativamente à metodologia.

3.6. Mais-Valias da metodologia DevOps

Para entender as mais-valias do DevOps torna-se também necessário compreender os resultados que permitem a concretização dessas mais-valias, bem como a forma como estas contribuem

para a adaptação das organizações a ambientes competitivos e voláteis no que diz respeito à inovação e à velocidade com que esta acontece.

Em termos de resultados na gestão diária do dia-a-dia das organizações, a metodologia DevOps propõe-se a alcançar os seguintes resultados:

- Ciclo de desenvolvimento mais curto[29];
- Maior velocidade de release[29];
- Melhor e maior deteção de erros[29];
- Menos falhas nos *deployments* e menos "*rollbacks*"[29];
- Redução do tempo necessário para recuperar de eventuais falhas[29];
- Aumento da colaboração e comunicação[29];
- Maior capacidade para investigação e inovação[29];
- Cultura orientada para o desempenho[29].
- Maior visibilidade e transparência sobre o ciclo de desenvolvimento[12].

Em geral, estes resultados traduzem-se num conjunto de mais-valias, a nível do negócio, para as organizações. Em baixo dá-se nota das mais-valias de negócio identificadas, considerando os resultados acima referidos:

- **Redução do tempo para o mercado**[13]: Ciclos de desenvolvimento mais curtos e maior velocidade na realização das *releases* possibilitam uma maior frequência de entradas em produção, que conferem a capacidade de desenvolver e entregar novas funcionalidades com maior rapidez e antecedência.

Por de trás da maior frequência, rapidez e segurança das entradas em produção estão as práticas DevOps, abordadas no capítulo 3.3, mas principalmente a automatização de processos, que lhes está associada.

Neste ponto, a tecnologia "Infrastructure as a Code" contribui também, como já aqui foi referido no capítulo 3.4, para uma aceleração do processo de desenvolvimento, permitindo uma maior rapidez e simplificação da gestão de ambientes para desenvolvimento, testes ou pré-produção, em conformidade com o de produção.

Esta mais-valia é bastante significativa para o negócio em diversas indústrias, principalmente em sectores de elevada concorrência e em que a exigência de inovação e rapidez de resposta ao mercado é elevada, podendo representar assim uma vantagem competitiva.

- **Feedback contínuo**[13]: Mais importante do que chegar ao mercado em primeiro, é chegar e entregar a solução mais adequada às necessidades e aspirações do utilizador final.

  Em resultado de uma maior comunicação e colaboração, não só com o cliente final mas também, e principalmente, entre as equipas de desenvolvimento e operações, é possível encontrar não só um foco maior e mais preciso nas necessidades do utilizador final, por um lado, como também uma maior coordenação interna.

  Assim, o feedback contínuo, abordado em maior detalhe no capítulo 3.3, é um ponto essencial da metodologia como condição necessária para a existência de uma organização ágil na adaptação às alterações de mercado e do meio competitivo em que se insere.

- **Um balanço melhorado entre custos e qualidade**[13]: Esta melhoria poderá dever-se a vários fatores, sendo que podemos apontar a deteção mais precisa de erros antes da entrada em produção, a maior comunicação e colaboração entre as equipas Dev e Ops ou a maior disponibilidade por parte dos *developers* para atividades de criação e inovação, como algumas das causas principais.

A automação permite também a estabilização de processos e, por exemplo, a realização de testes que eventualmente, fruto de imperativos temporais ou monetários, poderiam não ser realizados tendo em conta o tempo e custo que estes por vezes representam. A automação é também um fator de maior confiança na conformidade da solução, já que nenhum passo do processo é deixado em falso.

Ao automatizar processos podemos retirar o foco dos *developers* das tarefas repetitivas, longas e de baixo valor acrescentado, automatizando-as e redirecionando o sei foco para aquelas que garantam a qualidade e inovação das soluções desenvolvidas, e que por isso têm associado um valor acrescentado significativamente superior.

Aliado a esta maior disponibilidade para tarefas no âmbito da garantia de qualidade, a automação permitir permitir também uma maior racionalização dos recursos existentes, humanos inclusive, representando assim uma melhoria ao nível dos custos de desenvolvimento.

- **Uma maior previsibilidade das *releases***[13]: Tal como já fora referido no subcapítulo 3.3, a implementação de práticas *continuous integration*, *continuous delivery* ou *continuous monitoring*, permitem que tarefas como a integração de código, sucessivos testes e monitorização de todo o ciclo de desenvolvimento e correspondente fluxo, seja feito continuamente, permitindo antecipar eventuais erros cuja deteção tardia poderia representar um risco significativo para todo o desenvolvimento. No entanto, é importante ter sempre em mente que tal se deve principalmente à automação destes processos.

  Ainda assim, é essencial dar nota da importância que uma maior comunicação e colaboração entre as equipas Dev e Ops tem neste ponto. Esta maior coordenação permite assegurar que a *release* não só é atingível dentro do período estipulado, como também estará efetivamente preparada para entrar em produção, sem que com isso existam problemas relacionados com o processo de *deployment*, como por exemplo questões relacionadas com as configurações do ambiente de produção, entre outras.

  A existência de previsibilidades sobre aquilo que é produzido é preponderante para qualquer tipo de negócio ou indústria. Quanto maior for a previsibilidade, mais fácil será a

gestão de toda a empresa. Isto verifica-se tanto interna, na gestão de recursos técnicos ou humanos, como externamente, com uma redução significativa do risco de falha de entrega ao cliente final e uma maior liberdade sobre o modo como e quando a entrada em produção deverá ocorrer.

- **Aumento global da eficiência da organização**[13]: Após a identificação das mais-valias anteriores, é facilmente percetível o porquê de um aumento da eficiência. Quando aumentamos a rapidez dos processos de desenvolvimento e ao mesmo tempo garantimos a qualidade e a manutenção, senão mesmo a redução, de custos estamos inerentemente a garantir uma maior eficiência global.

  As razões que permitem alcançar este incremento de eficiência estão assim diretamente ligadas aos resultados já enunciados atrás para as mais-valias referidas. Este aumento de eficiência representa assim, mais uma vez, uma vantagem competitiva que o DevOps permite alcançar.

- **Redução do risco de negócio**[30]: Tendo em conta todos os resultados e correspondentes mais-valias já aqui abordadas, a redução do risco de negócio está já latente. Face à maior previsibilidade das *releases*, à maior deteção de erros ao longo do processo de desenvolvimento ou à sua maior transparência e visibilidade, o risco de negócio é consequentemente reduzido.

  Também o risco de chegarmos ao dia de "*go live*" e termos problemas relacionados com o processo de *deployment* - com as configurações relacionadas com o ambiente de produção ou com funcionalidade efetiva do que fora desenvolvido - é francamente reduzido. Prova disso são os resultados a que o DevOps se propõe como a redução de problemas de *deployment* e eventuais "*rollbacks*" ou a diminuição do tempo de recuperação após eventual falha.

Capítulo 4 - Demonstração e Análise de Resultados

Após a pesquisa e consequente estudo da metodologia DevOps, efetuada no capítulo anterior, o presente capítulo pretende dar a conhecer o estudo efetuado a um conjunto de casos de estudo onde foram implementadas soluções no âmbito da mesma.

Posto isto, o objetivo é o de procurar identificar os principais resultados obtidos da implementação da metodologia, para que posteriormente possam ser analisados em comparação com as mais-valias identificadas na secção 3.6.

Assim sendo, será aqui demonstrada a caracterização da amostra - secção 4.1 -, a metodologia de recolha e análise dos dados - secção 4.2 -, a demonstração dos resultados obtidos da análise aos casos de estudo – secção 4.3 – , consequente análise face às melhorias apontadas no capítulo 3 – secção 4.4 – e uma análise global dos resultados – secção 4.5.

4.1. Caracterização da Amostra

No âmbito do estudo efetuado, foram avaliados 18 casos de estudo, no âmbito da implementação de soluções tecnológicas, suportados na metodologia DevOps ou suas práticas e tecnologias. Estes casos de estudo, correspondem a "*white papers*" comerciais de fornecedores de referência nesta área, e estão distribuídos da seguinte forma:

Fornecedor de Serviços	Nº de Casos de Estudo
IBM	4
Microsoft	1
Puppet	5
Stelligent Systems LLC	3
XebiaLabs Customer Case Study	6

Tabela 2 - Distribuição dos casos de estudo pelos fornecedores proporietários.

No que diz respeito aos clientes onde as referidas soluções foram implementadas, existe uma maior presença de clientes provenientes dos Estados Unidos da América (11 no total), sendo imediatamente seguida por países da Europa, que em conjunto perfazem um total de 6 dos 18 casos de estudo.

País	Nº de Casos de Estudo
USA	11
Bélgica	1
França	1
Holanda	2
Nova Zelândia	1
UK	2

Tabela 3 - Distribuição dos casos de estudo por país.

No que refere à distribuição dos clientes pelas diferentes indústrias em que estão inseridos, existe, como já seria espectável, uma presença maior de empresas cujo negócio se baseia no desenvolvimento de *software* ou na prestação de serviços através do mesmo (referido apenas como indústria de software na tabela em baixo):

Indústria	Nº de Casos de Estudo
Seguros	3
Software	7
Entretenimento	1
Telecomunicações	1
Bancos	2
Companhias Aéreas	1
Energia	1
Serviços e Produtos de Imagem de Satélite	1
Retalho	1
Proteção Civil	1

Tabela 4 - Distribuição dos clientes dos casos de estudo pela indústria pertencente.

Ainda assim, existe também uma forte presença de empresas da indústria de serviços financeiros, em especial bancos e seguradoras, com um total de 4 casos de estudo. Esta forte presença evidencia a tendência cada vez maior, da indústria financeira, na aposta das plataformas digitais não só para a gestão interna de processos como também na sua interação com o cliente, através da adoção das plataformas móveis[31].

No que toca às soluções implementadas nos diferentes casos de estudo, foram identificados os seguintes conjuntos de soluções:

Soluções Implementadas	Nº de Casos de Estudo
DevOps	6
Continuous Delivery	5
Continuous Deployment	2
Infrastructure as a Code	1
Continuous Delivery & IAAC	2
DevOps & IAAC	2

Tabela 5 - Soluções implementadas nos casos de estudo.

Como se pode constatar, a generalidade dos casos de estudo não se referem à implementação da metodologia em todas as suas vertentes. As soluções denominadas de "DevOps", compreendem todos os casos de estudo em que se faz menção à metodologia DevOps enquanto solução implementada.

Todas as restantes soluções implementadas correspondem, tal como verificámos no capítulo 3, a componentes incluídas no âmbito DevOps, sendo também identificadas consoante a referência dada por cada caso.

No Anexo V, no final do presente estudo, é possível obter-se uma lista dos casos de estudo considerados.

4.2. Metodologia de recolha de dados

O objetivo central da análise dos casos de estudo recolhidos é o de identificar os resultados reportados em consequência da implementação de soluções enquadradas no âmbito da metodologia DevOps.

Os casos de estudo recolhidos são, somo já referido, "*papers*" comerciais dos diversos fornecedores de serviços, identificados na caracterização da amostra. Estes foram encontrados

nos *websites* dos respetivos proprietários, tendo sido selecionados aqueles cujas soluções se encontravam no âmbito da metodologia DevOps.

4.2.1. Recolha de dados

Após a seleção dos casos de estudo, a metodologia adotada para a recolha dos dados, teve por base a sua leitura e consequente identificação dos resultados reportados.

De modo a obter-se uma matriz de resultados comum a todos os casos, foi sendo elaborada uma tabela com a listagem dos casos de estudo e com os resultados. Posteriormente foram assinalados nas respetivas colunas, para cada um dos casos de estudo, os resultados identificados. À medida que os casos iam sendo lidos, os resultados identificados que ainda não se encontrassem nas colunas da tabela, ou que não se enquadrassem nos já existentes, eram adicionados.

Ao verificar a tabela em anexo, é possível constatar que foram identificados um total de 24 resultados diferentes, que foram agrupados dentro dos seguintes *clusters*:

- **Gestão de *Deployments*:** Agrupa todos os resultados referentes aos *deployments* ao longo do ciclo de desenvolvimento. De notar que, aqui, o conceito de *deployment* não se refere apenas à entrada em produção mas também noutros ambientes, tais como de desenvolvimento, teste e outros;

- **Gestão de Infraestrutura:** Aqui são incluídos os resultados obtidos no âmbito da gestão de infraestrutura, tais como: uma maior capacidade de gestão de infraestrutura ou a capacidade de provisionar mais rapidamente novos ambientes;

- **Gestão de *Releases*:** Este *cluster* agrupa todos os resultados que estejam relacionados com a gestão de *releases*, como por exemplo: maior qualidade, previsibilidade, rapidez na sua construção, entre outros;

- **Qualidade:** Este *cluster* acaba apenas por incorporar um único resultado, que é a constatação da melhoria da qualidade de código produzido;

- **Gestão e Controlo:** Dentro deste *cluster* estão agrupados todos os resultados relativos à gestão e controlo de processos e atividades incluídas no ciclo de desenvolvimento;

- **Feedback:** Aqui foram incluídos os resultados no âmbito da existência de feedback, não só por parte dos utilizadores finais como também entre diferentes equipas no ciclo de desenvolvimento;

- **Comunicação:** No âmbito deste *cluster* foram incluídos resultados que evidenciem melhorias efetivas na comunicação e partilha de conhecimento entre as diferentes equipas, com participação no ciclo de desenvolvimento;
- **Negócio:** Por último, todos os resultados relacionados com o impacto no negócio, tais como reduções de custo ou diminuição do risco operacional.

4.2.2. Metodologia de Análise aos Resultados

A análise que se pretende elaborar, com o estudo dos resultados reportados nos diferentes casos, tem como principal objetivo o de responder à questão principal do presente estudo, levantada no capítulo 1 e transcrita em baixo:

- Os benefícios aos quais a metodologia DevOps se propõe, são realmente verificados após a sua adoção?

Portanto, para responder à questão será feita uma análise interpretativa dos resultados obtidos face às mais-valias de negócio expectáveis provenientes da aplicação da metodologia DevOps, apontadas e descritas em pormenor no capítulo 3.6. O objetivo portanto é a verificação de resultados efetivos em casos reais e perceber se os mesmos correspondem às vantagens apontadas como sendo decorrentes da aplicação da metodologia DevOps.

A análise diz-se interpretativa, dado não existir uma metodologia estanque para suportar a mesma, sendo esta resultado da interpretação do autor do presente estudo. No entanto, a análise será levada a cabo de forma individual a cada mais-valia de negócio expectável, pela mesma ordem com que estas estão dispostas no capítulo 3.6.

4.3. Demonstração de Resultados

Tal como já vimos na caracterização da amostra, os casos de estudo estão distribuídos ao longo de um conjunto de 4 soluções diferentes, sendo que existem casos onde existe a implementação conjunta de duas soluções.

Na tabela 5, atrás, foram apresentadas as diferentes combinações de soluções implementadas nos diversos casos de estudo. Em baixo na tabela 6 são apresentados o número de casos por solução.

Solução	Nº de casos
Infrastructure as a Code	5
DevOps	8
Continuous Deployment	2
Continuous Delivery	7

Tabela 6 - Número de casos em que cada solução foi implementada.

A demonstração de resultados será feita por solução. Isto significa que em soluções que tenham pelo menos um caso de estudo em comum com outra solução diferente, os resultados demonstrados para ambas serão mutuamente influenciados. Tendo em conta os resultados espectáveis de cada uma das soluções, com base na informação constante no capítulo 3, será possível explicar eventuais resultados que poderão não ser expectáveis em determinadas soluções e que sejam fruto da influência dos casos de estudo que partilham.

Na página seguinte, são demonstrados os resultados numa tabela, de modo global. As colunas correspondem às diferentes soluções identificadas e as linhas aos *clusters*. Para cada solução é atribuída uma percentagem a cada *cluster*, que corresponde à percentagem de casos, referentes à solução, em que foram observados resultados dentro do *cluster* em causa. Por exemplo, para a solução "DevOps" e *cluster* "Gestão de *Deployments*", a percentagem é de 88%, ou seja dos casos em que foi implementada uma solução DevOps verificou-se que em 88% desses foram reportados resultados no âmbito da gestão de *deployments*.

Cluster	Solução			
	DevOps	IAAC	Continuous Delivery	Continuous Deployment
Gestão de Deployments	88%	60%	71%	100%
Gestão de Infraestrutura	63%	100%	71%	0%
Gestão de Releases	63%	40%	57%	0%
Qualidade de Código	25%	0%	29%	50%
Gestão e Controlo	38%	80%	57%	50%
Feedback	50%	20%	29%	0%
Comunicação	63%	60%	14%	0%
Negócio	75%	100%	71%	100%

Tabela 7 - Tabela de demonstração global de resultados

Nas secções seguintes, serão endereçadas cada uma das soluções, retiradas as primeiras impressões, bem como identificados os anexos correspondentes para análise pormenorizada dos resultados obtidos em cada *clust*

4.3.1. DevOps

Solução	Cluster							
	Gestão de Deployments	Gestão de Infraestrutura	Gestão de Releases	Qualidade de Código	Gestão e Controlo	Feedback	Comunicação	Negócio
DevOps	88%	63%	63%	25%	38%	50%	63%	75%

Tabela 8 - Resultados por cluster para a solução DevOps.

Ao olhar para os resultados reportados nos casos de estudo onde foi implementada uma solução DevOps, podemos constatar que o maior número de resultados se encontra dentro da gestão de *deployments* e de negócio.

Conforme pode ser observado no Anexo I, existiu dentro da gestão de *deployments* um resultado específico que fora citado na larga maioria dos casos: uma redução do tempo por *deployment*, ou se quisermos *deploys* mais rápidos. Pode-se concluir que esta maior rapidez se deverá à automação destes processos. Como já vimos, automação é referida como sendo capaz de retirar a intervenção humana dos processos, permitindo que estes sejam mais estáveis, eliminando o risco de erro humano e levando assim a uma maior confiança, fiabilidade (menos erros/falhas) e rapidez do processo.

Ao nível dos resultados obtidos no âmbito do negócio, existe um que se distingue dos restantes: a disponibilização de mais tempo para tarefas de maior valor acrescentado. Tal observação foi repetidamente referida como sendo consequência da automação dos sucessivos processos, ou *delivery pipeline*, que permite eliminar atividades longas, repetitivas e que muitas vezes correspondem a atividades de reduzido valor acrescentado.

Para além dos dois *clusters* citados, existe ainda uma percentagem significativa, de resultados reportados no âmbito dos seguintes clusters: Gestão de Infraestrutura, Gestão de *Releases*, Comunicação e Feedback.

No que diz respeito à Gestão de Infraestrutura, tendo em conta que a solução DevOps partilha dois dos seus seis casos de estudo com a solução de *Infrastructure as a Code,* onde se verificaram resultados neste cluster em 100% dos casos (ver resultados para a solução *Infrastructure as a Code* e *cluster* de gestão de infraestrutura na tabela 7), pode-se afirmar que o valor obtido (de 63%), é em certa medida influenciado pelos casos em comum entre ambas as soluções. No entanto, tendo em conta que a tecnologia *Infrastructure as a Code* é referida como sendo uma tecnologia DevOps, capítulo 3.4, estes resultados podem ser perfeitamente interpretados como resultados da solução.

Analisando os resultados obtidos no âmbito da comunicação, o resultado mais frequentemente citado fora o da melhoria da comunicação entre as diferentes equipas participantes no ciclo desenvolvimento (ver anexo I), com principal foco na comunicação entre as equipas de desenvolvimento e operações, estando assim em linha com a atenção do DevOps nestas duas equipas (capítulo 3.1).

A percentagem de 63% dos casos com resultados reportados no que refere à gestão de *releases*, resulta essencialmente da verificação de uma maior frequência com que as *releases* são realizadas/finalizadas, bem como a sua maior rapidez e qualidade (ver anexo I).

No que toca aos resultados obtidos no âmbito do Feedback, nos 50% dos casos onde foram reportados, verificou-se um aumento da rapidez com que este passou a ser obtido e na adoção das necessidades do utilizador final (ambos 38% dos 50%), bem como uma melhoria da qualidade do feedback recebido (25% dos 50%).

Por fim, no que diz respeito aos *clusters* de Qualidade (com resultados reportados em 25% dos casos) e de Gestão e Controlo (38%) os únicos resultados reportados foram a melhoria da qualidade de código, para o primeiro, e uma maior consciência e visibilidade sobre os processos.

4.3.2. Infrastructure as a Code

Solução	Cluster							
	Gestão de Deployments	Gestão de Infraestrutura	Gestão de Releases	Qualidade de Código	Gestão e Controlo	Feedback	Comunicação	Negócio
IAAC	60%	100%	40%	0%	80%	20%	60%	100%

Tabela 9 - Resultados por cluster para a solução Infrastructure as a Code.

Os dados reportados no âmbito da implementação de soluções de *Infrastructure as a Code* vão, globalmente, de encontro ao expectável, no que aos *clusters* em que se verificaram resultados, diz respeito.

Os *clusters* "Gestão de Infraestrutura" e " Negócio" obtiveram resultados em todos os casos de estudo em que a solução foi implementada. No que refere à gestão de infraestrutura, é de salientar que em 80% dos casos de estudo é referido um aumento na rapidez com que os ambientes são provisionados. Já no que diz respeito ao negócio, o principal resultado foi a libertação de mais tempo para tarefas de maior valor acrescentado, também verificado em 80% dos casos.

Em seguida, em termos de percentagens, surge o *cluster* de gestão e controlo com 80% e a gestão de *deployments* e comunicação, ambos com 60%. A maior consciência e visibilidade sobre os processos, verificada em 60% dos casos de estudo, surge como o resultado mais frequente dentro do cluster de gestão e controlo, podendo facilmente explicar-se pela harmonização de ambientes, e facilidade em proceder-se a alterações nos mesmos, que esta tecnologia permite.

Já no que refere à gestão de *deployments*, o resultado mais frequentemente reportado é o da redução do tempo médio por *deployment*, verificado em 60% dos casos, seguido pelo seu aumento (frequência) e fiabilidade, ambos verificados em 40% dos casos. Estes resultados ao nível da gestão de *deployments* podem ser facilmente explicados, mais uma vez, pela harmonização de ambientes, que permite que o código desenvolvido seja desenvolvido e testado em ambientes o mais semelhantes possível ao de produção, resultando assim num menor número de erros e problemas relacionados com adaptações de ambiente.

Ao nível da comunicação, os resultados estão em linha com os dados apresentados no capítulo 3, onde é referida uma melhoria significativa da colaboração entre as equipas de desenvolvimento e operações (ver figura 6 – secção 3.4). Isto porque o resultado que se destacou foi a melhoria da comunicação entre as diferentes equipas envolvidas, verificado em 60% dos casos.

No que respeita à gestão de releases, foi verificado uma maior frequência com que as mesmas são efetuadas em 20% dos casos, acabando por estar pouco abaixo do valor apresentado pelo

estudo da Forrester referido no subcapítulo 3.4 (ver figura 6), onde este resultado é apontado por 36% e 26% dos profissionais de desenvolvimento e operações, respetivamente.

Por fim, não foram verificados resultados ao nível da qualidade de código e a existência de resultados obtidos ao nível do feedback corresponde apenas a um caso de estudo, em que também é implementada uma solução de *Continuous Delivery*, sendo que não é considerado aqui significativo para a solução de *Infrastructure as a Code*.

4.3.3. Continuous Delivery

Solução	Cluster							
	Gestão de Deployments	Gestão de Infraestrutura	Gestão de Releases	Qualidade de Código	Gestão e Controlo	Feedback	Comunicação	Negócio
Continuous Delivery	71%	71%	57%	29%	57%	29%	14%	71%

Tabela 10 - Resultados por cluster para a solução Continuous Delivery.

A nível da solução de *continuous delivery*, aplicada em 7 dos casos estudados, os resultados nos *clusters* de gestão de *deployments*, infraestrutura e de negócio são os que prevalecem com uma maior percentagem de casos de estudo, 71%.

No que diz respeito à gestão de *deployments*, os resultados de redução do tempo por *deployment* e o aumento da fiabilidade foram os mais verificados, ambos apontados em 57% dos casos, em linha com o expectável face à descrição da prática no subcapítulo 3.3. Já no que diz respeito à gestão da infraestrutura, o resultado mais citado foi o de uma maior capacidade de gestão da mesma, verificado em 57% dos casos, e no que refere ao negócio constatou-se mais tempo para tarefas de maior valor acrescentado que surge, a par da redução custos, com uma percentagem de 71% dos casos.

Após estes três *clusters*, surgem os clusters de gestão de *releases* e o de gestão e controlo, ambos com resultados verificados em 57% dos casos. *Releases* mais rápidas e maior previsibilidade nas mesmas são os resultados mais verificados no primeiro, em 43% dos casos, sendo que no cluster de gestão e controlo o resultado com maior ênfase foi o de uma maior consciência e visibilidade

sobre os processos, verificado em 43% dos casos. Tais resultados coincidem com as expectativas criadas no capítulo 3.

Nos restantes *clusters*, o resultado mais significativo verificou-se ao nível da qualidade, com a verificação de uma maior qualidade do código desenvolvido, em 29% dos casos.

4.3.4. Continuous Deployment

Solução	Cluster							
	Gestão de Deployments	Gestão de Infraestrutura	Gestão de Releases	Qualidade de Código	Gestão e Controlo	Feedback	Comunicação	Negócio
Continuous Deployment	100%	0%	0%	50%	50%	0%	0%	100%

Tabela 11 - Resultados por cluster para a solução Continuous Deployment.

Nos dois casos de estudo recolhidos em que foram implementadas soluções de *continuous deployment*, apenas se verificaram resultados em quatro dos oito *clusters*: na gestão de *deployments*, qualidade de código, gestão e controlo e a nível do negócio.

Tendo em conta o número reduzido de casos, é importante frisar apenas os resultados verificados em ambos, que se verificaram nos *clusters* de gestão de *deployments* e de negócio. A nível da gestão de *deployments*, nenhum dos casos referiu nenhum resultado relacionado com a qualidade do processo, tendo ambos verificado uma maior fiabilidade do processo, que no fundo se traduz em *deployments* mais estáveis e com menos erros.

Já no que refere ao negócio, apenas um resultado foi verificado em ambos os casos, que foi a libertação de mais tempo, por parte dos diferentes intervenientes no ciclo de desenvolvimento, para tarefas de maior valor acrescentado. Este resultado é essencialmente fruto da automação dos processos de *deployment* repetitivos e grandes consumidores de tempo, permitindo, para além de libertar as equipas para tarefas de maior valor acrescentado, aumentar a rapidez dos *deployments* (resultado reportado num dos casos).

Já a verificação de resultados a nível da gestão, controlo e qualidade, resulta essencialmente da padronização e estabilidade que a automação permitiu conceder a estes processos e das

componentes de reporte sobre os mesmos, que as soluções fornecidas pelos fornecedores de serviços disponibilizam.

4.4. Análise dos Resultados

4.4.1. Redução do Tempo para o Mercado

Ao olharmos para a percentagem de casos, de qualquer uma das soluções, em que fora reportada uma redução do tempo de chegada ao mercado, poderíamos afirmar que esta mais-valia não se verificou significativamente. A solução onde este resultado ainda foi significativo, em percentagem, foi na solução de *Continuous Deployment* com um resultado de 50%, sendo que aqui pesa o facto de esta solução apenas representar 2 dos 18 casos considerados, tendo assim pouca relevância.

No entanto, torna-se importante ressalvar que o facto de este resultado não ter sido diretamente reportado pelos autores dos diferentes estudos não significa que este não se tenha verificado. Para perceber melhor se este resultado poderá ter tido uma maior verificação, torna-se necessário perceber as percentagens obtidas pelos resultados subsequentes que permitem às organizações diminuir o tempo de chegada ao mercado.

Tal como já foi referido no capítulo 3.6, uma maior frequência das *releases* bem como das entradas em produção poder-se-ão considerar fatores essenciais para a verificação efetiva da redução do tempo de chegada ao mercado. Assim, ao olharmos para a percentagem verificada nestes resultados constatamos o seguinte cenário:

Resultado Reportado	DevOps	IAAC	C. Delivery	C. Deployment
Aumento do número de *deployments*	38%	40%	29%	50%
Redução do tempo por *deployment*	75%	60%	57%	50%
Releases mais rápidas	38%	20%	43%	0%
Maior frequência das *releases*	50%	20%	29%	0%

Tabela 12 - Resultados no âmbito da gestão de releases para as diferentes soluções.

Aqui podemos perceber que a solução DevOps obteve, de um modo geral, percentagens superiores face às restantes, e bastante significativas (variam entre os 38% - 75%), principalmente na redução do tempo por *deployment* (75%), o que se traduz numa maior velocidade no processo

de entrada em produção e também influência positivamente a frequência com que as *releases* são lançadas (50%).

Posto isto, pode-se assim concluir que uma solução DevOps demonstrou efetivamente resultados que poderão levar à redução do tempo de chegada ao mercado.

4.4.2. Feedback Contínuo

Para analisar os resultados obtidos face à verificação, ou não, de feedback contínuo como resultado da implementação da metodologia DevOps, torna-se necessário olhar para os resultados obtidos no âmbito dos clusters de feedback e comunicação:

Resultado Reportado	DevOps	IAAC	C. Delivery	C. Deployment
Maior rapidez na adoção das necessidades identificadas dos clientes.	38%	20%	14%	0%
Feedback mais rápido	38%	0%	14%	0%
Melhores insights por parte do cliente	25%	0%	0%	0%
Melhor comunicação/colaboração entre as diferentes equipas	63%	60%	14%	0%
Processos de comunicação mais eficientes	13%	20%	0%	0%

Tabela 13 - Resultados obtidos no âmbito do feedback e comunicação nas diversas soluções.

Aqui, mais uma vez podemos ver que os resultados são na generalidade melhores na solução DevOps, onde é apresentada uma verificação em 38% dos seus casos de uma maior rapidez do feedback recebido e, inclusive, uma melhor qualidade desse feedback por parte do utilizador final, verificado em 25% dos casos.

4.4.3. Balanço Melhorado entre Custos e Qualidade

Nos resultados obtidos não existe nenhuma referência direta a esta melhoria do balanço entre custo e qualidade.

Ao nível dos custos, apenas um caso DevOps refere a sua redução. No entanto este resultado não condiz com as percentagens verificadas nas restantes soluções: 40% dos casos IaaC e 71% dos casos *Continuous Delivery*.

Assim, face a este paralelismo de resultados não é possível concluir nada em concreto, no que aos custos diz respeito, já que se por um lado os casos em que foi implementada uma solução DevOps não foi reportada, significativamente, uma redução dos custos, por outro as restantes soluções que estão no seu âmbito já apresentam.

No que diz respeito à melhoria da qualidade, apenas 25% dos casos DevOps referem uma melhoria de qualidade, em linha com os 29% dos casos de *Continuous Delivery*, não sendo ainda assim um valor muito, nem pouco, significativo para retirar conclusões.

Posto isto, não é possível afirmar com segurança que uma solução DevOps resulta, ou não, num balanço melhorado entre o custo e qualidade, dado os resultados serem pouco significativos e inconclusivos.

4.4.4. Maior previsibilidade nas *Releases*

Neste ponto em particular, nenhum caso em que fora implementada uma solução DevOps se referiu diretamente à existência ou não de uma maior previsibilidade nas *releases*.

No entanto, este resultado verificou-se em 43% dos casos de *Continuous Delivery*, sendo que estando esta prática incluída no âmbito da metodologia DevOps, seria expectável que os resultados em ambas as soluções estivessem em maior concordância.

Tendo isto em conta, não é possível auferir nenhuma conclusão acerca do impacto, ou não, da metodologia DevOps na previsibilidade das *releases*.

4.4.5. Aumento Global da Eficiência da Organização

No que toca a uma eventual melhoria global da eficiência organizacional, os diferentes casos de estudo não referem diretamente a questão.

No entanto, tendo em conta não terem sido reportados aumentos de custos em nenhum dos casos de estudo, de se ter verificado uma diminuição, por exemplo, no número de recursos humanos necessários para a gestão de *deployments* (em 25% dos casos DevOps, e em 14% dos

casos de *Continous Delivery*), de se ter verificado resultados significativos a nível da rapidez tanto dos *deployments* como da próprias *releases*, um maior feedback e comunicação ao longo do ciclo de desenvolvimento e, por fim, uma maior disponibilidade para tarefas de maior valor acrescentado, é plausível afirmar que a metodologia DevOps, e consequentes práticas e tecnologias, contribuem para um aumento da eficiência das organizações.

4.4.6. Redução do Risco de Negócio

Ao nível do risco de negócio, foi verificada uma redução do risco operacional em 25% dos casos que envolvem uma solução DevOps, 40% dos casos que envolvem uma solução IaaC e em 14% dos casos de *Continuous Delivery*.

Tais resultados refletem que existe de facto alguns resultados positivos ao nível do risco de negócio, que a meu ver é fruto de vários outros resultados (apenas alguns exemplos):

- Ao nível de uma maior consciência e visibilidade dos processos, verificado em 38% dos casos que envolvem uma solução DevOps, 60% dos casos que envolvem uma solução IaaC e em 43% dos casos de *Continuous Delivery*;

- Ao nível da gestão de infraestrutura, cujos resultados, tanto na rapidez e eficiência de provisionamento como na capacidade de gestão de infraestrutura, foram globalmente satisfatórios em todas em soluções, com especial enfoque na solução DevOps mas principalmente na IaaC;

- Na maior fiabilidade dos *deployments*, resultado que obteve também resultados significativos, refletindo uma maior confiança nos processos de *deployment*.

Ainda que não existam resultados que reflitam diretamente uma redução do risco operacional, os três pontos atrás referidos podem em conjunto, ou separadamente, ter como consequência essa mesma redução .

4.5. Análise Global

Já aqui foi feita uma análise aos resultados face às mais-valias apontadas no capítulo 3. No entanto considerou-se importante fazer uma reflecção desprendida dessas mais-valias e sobre uma perspetiva global.

Na demonstração de resultados foram já apresentados os resultados mais significativos para todas as soluções sendo que gostaria aqui de dar nota dos seguintes:

- Todas as soluções apresentam resultados significativos no que respeita à redução do tempo por *deployment* – em todas com percentagens de constatação igual ou superior a 50%. Tal indicador reflete uma maior rapidez dos processos de desenvolvimento e entrada em produção;

- Os resultados ao nível da gestão de infraestruturas, ao longo das diferentes soluções abordadas, são também eles bastante significativos. Com especial enfoque na solução IaaC, o que seria espectável dado ser o principal foco desta tecnologia, verificaram-se bons resultados ao nível da rapidez e eficácia na gestão e controlo da infraestrutura;

- Os casos em que foram implementadas soluções DevOps refletiram resultados significativos ao nível da melhoria da comunicação entre as diferentes equipas ao longo do ciclo de desenvolvimento. Este resultado está em linha com a preocupação da metodologia DevOps em promover uma maior colaboração entre as diferentes equipas envolvidas ao longo do ciclo de desenvolvimento.

- Um resultado que foi certamente uma constante na grande maioria dos casos de estudo, independentemente da solução, foi o da verificação de uma maior disponibilidade por parte dos *developers* para tarefas de maior valor acrescentado. Tal resultado é frequentemente apontado como sendo fruto da automação de processos repetitivos, longos e de baixo valor acrescentado. A verificação deste resultado contribui em muito para uma gestão mais eficiente dos recursos humanos e, consequentemente, para uma maior eficiência global das organizações dado que estes podem agora focar-se mais em tarefas que primem pela qualidade ou inovação.

Capítulo 5 - Conclusões

5.1. Principais Conclusões

No que respeita ao conceito da metodologia DevOps, concluiu-se que o seu objetivo principal é o de transformar soluções desenvolvidas em valor efetivo o mais rapidamente possível, procurando assim responder a uma necessidade que a metodologia Agile não respondia. Para isso esta foca-se na relação entre as equipas de desenvolvimento e operações como meio de garantir uma maior conformidade dos desenvolvimentos para a entrada em produção, passando também a implicar ou a dar mais atenção aos seguintes pontos:

- Mudança cultural focada na celebração do erro, na perspetiva de que quanto mais cedo estes forem detetados menor deverá ser o impacto que estes terão no que já foi ou estará para ser desenvolvido;

- Automatização do maior número de processos repetitivos e de baixo valor acrescentado, dando assim aso a um maior foco em atividades que promovam a inovação e qualidade dos desenvolvimentos;

- Na adoção de ciclos de feedback, planeamento e monitorização contínuos promovendo deste modo uma maior agilidade das organizações na adaptação e acompanhamento das alterações de mercado;

Foi possível também concluir que a metodologia DevOps pressupõe a aplicação da metodologia Agile, não vindo assim substituí-la, e aproveita já um conjunto existente de práticas de desenvolvimento ao longo de todo o ciclo de desenvolvimento para não só torna-lo mais ágil, como também mais eficiente.

No que respeita aos benefícios de negócio que é possível alcançar através da sua implementação, foram levantadas 6 mais-valias durante a revisão bibliográfica, que no entanto não foi possível verifica-las com correspondência e significância suficiente para responder à principal questão do estudo. Assim sendo, não é possível afirmar negativa nem positivamente em relação à correspondência entre os resultados propostos da metodologia, levantados na revisão bibliográfica, e os resultados obtidos nos diferentes casos de estudo.

No entanto, importa salientar os principais resultados obtidos na análise global. Com especial enfoque nas soluções DevOps, foram obtidos resultados muito significativos no controlo e gestão de infraestrutura, na rapidez e fiabilidade de processos, na comunicação entre diferentes equipas e verificou-se mais tempo para dedicar a tarefas de inovação e garantia de qualidade.

Todos estes resultados observados representam vantagens competitivas significativas para as organizações. Ao melhorar a gestão e controlo da infraestrutura de TI existente, poderá representar não só uma racionalização de custos com a mesma, como também uma maior fiabilidade das soluções desenvolvidas, tendo em conta a uniformização de ambientes.

Já a vertente de automação da metodologia DevOps assume um papel central na maior rapidez e fiabilidade de sucessivos processos ao longo do ciclo de desenvolvimento. Ao automatizar está-se a conceder uma uniformização dos processos, um incremento na sua rapidez e a retirar a intervenção humana dos mesmos. Tudo isto leva a um redução de custos, garantia de qualidade e maior rapidez das organizações, tão importante no ambiente competitivo da economia atual.

No que refere à comunicação, muitas são as vantagens que as organizações retiram da sua melhoria, já que permite a partilha de conhecimento, diminuindo tempos de produção ao prevenir a duplicação de trabalho e contribuindo para uma maior fiabilidade e qualidade do que é desenvolvido.

Por fim, ao verificar-se mais tempo para tarefas de inovação e garantia de qualidade a metodologia DevOps permite às organizações melhorar a forma como os recursos são aproveitados, reencaminhando-os para tarefas de real valor acrescentado.

5.2. Contributos e Limitações do Estudo Realizado

O presente estudo permitiu trabalhar uma visão global da metodologia, procurando encontrar correspondência face às referências bibliográficas nos casos de estudo recolhidos que, ainda que não tenha sido significativa, permitiu identificar tendências claras quanto às mais-valias que esta pode trazer às organizações.

Como principais limitações ao estudo efetuado, contam-se os seguintes pontos:

- As poucas referências científicas encontradas, referentes ao tema em estudo;
- O cariz de "*white papers*" comerciais dos casos de estudo recolhidos, que de algum modo poderá por em causa a credibilidade científica e foco nos resultados dos mesmos;
- O facto de os resultados poderem ser influenciados pelas características específicas de cada solução implementada em cada caso de estudo;
- A inexistência de casos de estudo que tenham por base o mesmo método de implementação da metodologia e a mesma interpretação do seu conceito e abrangência.

5.3. Sugestões para Investigação Futura

No que respeita ao futuro, torna-se primeiro essencial encontrar um consenso comum, à volta da comunidade ligada ao desenvolvimento de *software*, de modo a definir um conceito estanque e princípios para a metodologia DevOps, tal como já hoje acontece com a metodologia Agile.

Tal consenso iria permitir a aplicação da metodologia de modo uniforme para vários casos, permitindo assim retirar o fator de subjetividade e concedendo uma maior fiabilidade e segurança, aos seus resultados.

Bibliografia

[1] The Standish Group, "The Standish group: the chaos report," *Proj. Smart*, p. 16, 2014.

[2] S. Sharma, *DevOps for Dummies*. 2014.

[3] D. . Womack, J.P. and Jones, *Lean Thinking*. 1997.

[4] M. Poppendieck and P. Llc, "Principles of Lean Thinking Origins of Lean Thinking," *System*, vol. D, no. July, pp. 1–7, 2002.

[5] V. Studio, B. D. J. Anderson, D. J. Anderson, D. Development, C. P. Improvement, T. Note, W. Not, E. Both, and L. S. Society, "Lean Software Development," 2013.

[6] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. 2003.

[7] D. K. Rigby, J. Sutherland, and H. Takeuchi, "The Secret History of Agile Innovation," *Harv. Bus. Rev.*, p. 7, 2016.

[8] K. Beck, M. Beedle, and A. Bennekum, "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed: 03-Mar-2016].

[9] K. Petersen, "Is Lean Agile and Agile Lean?: A Comparison between Two Software Development Paradigms," *IGI Glob. Chapter 2*, no. Beck 2000, pp. 19–46, 2011.

[10] I. H. Sarker, F. Faruque, U. Hossen, and A. Rahman, "A survey of software development process models in software engineering," *Int. J. Softw. Eng. its Appl.*, vol. 9, no. 11, pp. 55–70, 2015.

[11] B. BOEHM and R. TURNER, "Management challenges to implementing agile processes in traditional development organizations," *IEEE Comput. Soc.*, 2005.

[12] K. Bittner, A. Demartine, and D. Lo Giudice, "Use DevOps And Supply Chain Principles To Automate Application Delivery Governance," 2015.

[13] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," *5th Int. Conf. Innov. Comput. Technol. INTECH 2015*, no. Intech, pp. 78–82, 2015.

[14] C. A. Cois, "DevOps and Agile," 2014. [Online]. Available: https://insights.sei.cmu.edu/sei_blog/2014/11/devops-and-agile.html.

[15] W. I. Devops, "The agile admin Definition of DevOps," 2015. [Online]. Available: https://theagileadmin.com/what-is-devops/.

[16] M. Huntterman, *DevOps for Developers*. .

[17] T. Good, T. Bad, T. Ugly, O. Devops, P. One, and O. Three, "Brief : Good DevOps Requires Collaboration , Automation , And Cultural Change," 2015.

[18] W. Read, T. Report, and K. Takeaways, “DevOps Makes Modern Service Delivery Modern,” 2015.

[19] M. Fowler, “Continuous Integration,” 2006. [Online]. Available: http://www.martinfowler.com/articles/continuousIntegration.html. [Accessed: 15-May-2016].

[20] Agile Alliance, “Continuous Integration.” [Online]. Available: https://www.agilealliance.org/glossary/continuous-integration/. [Accessed: 15-May-2016].

[21] J. Humble, “Continuous Delivery vs Continuous Deployment,” 2010. [Online]. Available: https://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/. [Accessed: 04-Apr-2016].

[22] B. Fitzgerald and K.-J. Stol, “Continuous software engineering and beyond: trends and challenges,” *Proc. 1st Int. Work. Rapid Contin. Softw. Eng. - RCoSE 2014*, pp. 1–9, 2014.

[23] C. Caum, “Continuous Delivery Vs. Continuous Deployment: What’s the Diff?,” 2013. [Online]. Available: https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff. [Accessed: 20-May-2016].

[24] W. Read and T. Report, “Express Lean , Agile , And DevOps Cultural Changes In Terms Of Business Value,” pp. 1–6, 2014.

[25] IBM Corporation, “DevOps: The IBM approach,” pp. 1–12, 2014.

[26] K. Bittner, A. Demartine, D. Lo Giudice, and R. Heffner, “Use DevOps Practices To Create A Lean And Responsive Application Delivery Organization,” 2016.

[27] Forrester Research, “Infrastructure As Code: Fueling The Fire For Faster Application Delivery,” no. March, pp. 1–10, 2015.

[28] J. Bird, *Devops for Finance*, vol. 1. O’Reilly, 2015.

[29] M. Paul, “The Measurable and Important Benefits of DevOps.” [Online]. Available: http://www.logicworks.net/blog/2014/10/measurable-important-benefits-devops/. [Accessed: 26-Jun-2016].

[30] “DevOps : Delivering at the speed of today ’ s business What is DevOps ?” Accenture Architecture Services.

[31] WEF, “The Future of Financial Services - How disruptive innovations are reshaping the way financial services are structured , provisioned and consumed,” *Wef*, no. June, pp. 1–178, 2015.

Anexos

Anexo I

Tabela de resultados reportados pormenorizada – DevOps

	Resultado Reportado	%
Gestão de Deployments	Aumento do número de deployments	38%
	Redução do tempo por deployment	75%
	Maior fiabilidade dos deployments	25%
	Maior qualidade dos deployments	13%
	Redução de recursos humanos necessários	25%
Gestão de Infraestrutura	Maior capacidade de gestão da infraestrutura	38%
	Provisionamento mais rápido e eficaz de ambientes	50%
Gestão de Releases	Releases mais rápidas	38%
	Maior previsibilidade nas releases	0%
	Maior qualidade das releases	38%
	Maior frequência das releases	50%
Qualidade	Maior qualidade do código	25%
Gestão e Controlo	Maior consciência e visibilidade sobre os processos	38%
	Maior deteção de erros	0%
	Maior orquestração entre processos	0%
Feedback	Maior rapidez na adoção das necessidades identificadas dos clientes.	38%
	Feedback mais rápido	38%
	Melhores insights por parte do cliente	25%
Comunicação	Melhor comunicação/colaboração entre as diferentes equipas	63%
	Processos de comunicação mais eficientes	13%
Negócio	Mais tempo para tarefas de maior valor acrescentado	75%
	Redução do tempo para o mercado	13%
	Redução de custos	13%
	Diminuição do risco operacional	25%

Anexo II

Tabela de resultados reportados pormenorizada – Infrastructure as a Code

	Resultado Reportado	%
Gestão de Deployments	Aumento do número de deployments	40%
	Redução do tempo por deployment	60%
	Maior fiabilidade dos deployments	40%
	Maior qualidade dos deployments	20%
	Redução de recursos humanos necessários	0%
Gestão de Infraestrutura	Maior capacidade de gestão da infrastrutura	60%
	Provisionamento mais rápido e eficaz de ambientes	80%
Gestão de Releases	Releases mais rápidas	20%
	Maior previsibilidade nas releases	20%
	Maior qualidade das releases	20%
	Maior frequência das releases	20%
Qualidade	Maior qualidade do código	0%
Gestão e Controlo	Maior consciência e visibilidade sobre os processos	60%
	Maior deteção de erros	0%
	Maior orquestração entre processos	20%
Feedback	Maior rapidez na adoção das necessidades identificadas dos clientes.	20%
	Feedback mais rápido	0%
	Melhores insights por parte do cliente	0%
Comunicação	Melhor comunicação/colaboração entre as diferentes equipas	60%
	Processos de comunicação mais eficientes	20%
Negócio	Mais tempo para tarefas de maior valor acrescentado	80%
	Redução do tempo para o mercado	20%
	Redução de custos	40%
	Diminuição do risco operacional	40%

Anexo III

Tabela de resultados reportados pormenorizada – Continuous Delivery

	Resultado Reportado	%
Gestão de Deployments	Aumento do número de deployments	29%
	Redução do tempo por deployment	57%
	Maior fiabilidade dos deployments	57%
	Maior qualidade dos deployments	29%
	Redução de recursos humanos necessários	14%
Gestão de Infraestrutura	Maior capacidade de gestão da infraestrutura	57%
	Provisionamento mais rápido e eficaz de ambientes	29%
Gestão de Releases	Releases mais rápidas	43%
	Maior previsibilidade nas releases	43%
	Maior qualidade das releases	29%
	Maior frequência das releases	29%
Qualidade	Maior qualidade do código	29%
Gestão e Controlo	Maior consciência e visibilidade sobre os processos	43%
	Maior deteção de erros	14%
	Maior orquestração entre processos	14%
Feedback	Maior rapidez na adoção das necessidades identificadas dos clientes.	14%
	Feedback mais rápido	14%
	Melhores insights por parte do cliente	0%
Comunicação	Melhor comunicação/colaboração entre as diferentes equipas	14%
	Processos de comunicação mais eficientes	0%
Negócio	Mais tempo para tarefas de maior valor acrescentado	71%
	Redução do tempo para o mercado	14%
	Redução de custos	71%
	Diminuição do risco operacional	14%

Anexo IV

Tabela de resultados reportados pormenorizada – Continuous Deployment

	Resultado Reportado	%
Gestão de Deployments	Aumento do número de deployments	50%
	Redução do tempo por deployment	50%
	Maior fiabilidade dos deployments	100%
	Maior qualidade dos deployments	0%
	Redução de recursos humanos necessários	50%
Gestão de Infraestrutura	Maior capacidade de gestão da infraestrutura	0%
	Provisionamento mais rápido e eficaz de ambientes	0%
Gestão de Releases	Releases mais rápidas	0%
	Maior previsibilidade nas releases	0%
	Maior qualidade das releases	0%
	Maior frequência das releases	0%
Qualidade	Maior qualidade do código	50%
Gestão e Controlo	Maior consciência e visibilidade sobre os processos	50%
	Maior deteção de erros	0%
	Maior orquestração entre processos	0%
Feedback	Maior rapidez na adoção das necessidades identificadas dos clientes.	0%
	Feedback mais rápido	0%
	Melhores insights por parte do cliente	0%
Comunicação	Melhor comunicação/colaboração entre as diferentes equipas	0%
	Processos de comunicação mais eficientes	0%
Negócio	Mais tempo para tarefas de maior valor acrescentado	100%
	Redução do tempo para o mercado	50%
	Redução de custos	50%
	Diminuição do risco operacional	50%

Anexo V

Lista de Casos de Estudo

Empresa	Indústria	Data	Geografia	Fornecedor de Serviços
IBM Software Group	Software	2014	EUA	IBM
IBM	Software	2015	EUA	IBM
Nationwide	Seguros	2014	EUA	IBM
Sony Pictures Technologies	Entretenimento	n.a	EUA	Stelligent Systems LLC
Cable & Wireless	Comunicações	n.a	Reino Unido	Stelligent Systems LLC
ING Bank	Banca	n.a	Reino Unido	XebiaLabs Customer Case Study
Air France-KLM	Companhia Aérea	n.a	França	XebiaLabs Customer Case Study
Medecision	Software	n.a	EUA	XebiaLabs Customer Case Study
Ambit Energy	Energia	2013	EUA	Puppet
Rabobank	Banca	n.a	Holanda	XebiaLabs Customer Case Study
Stater	Seguros	n.a	Holanda	XebiaLabs Customer Case Study
Advent	Software	n.a	EUA	Stelligent Systems LLC
Nowcom	Software	2014	EUA	Microsoft
Geospatial Product and Service Provider Company	Serviços baseados em imagens de satélite	n.a	EUA	XebiaLabs Customer Case Study
Staples	Retalho	n.a	EUA	Puppet
GNS Science	Setor Público	n.a	Nova Zelândia	Puppet
Smarbit	Software	n.a	Bélgica	Puppet
IP Comerce	Software	n.a	EUA	Puppet